ioProgrammo

ANTEPRIMA JAVA 6

NOME IN CODICE MUSTANG, PRONTO PER IL DESKTOP. COSA CAMBIERÀ PER GLI SVILUPPATORI?

Rivista + "Le grandi guide di ioProgrammo" a € 12,90 in più

VERSIONE PLUS RIVISTA+LIBRO+CD €9,90 — VERSIONE STANDARD RIVISTA+CD €6,90

PER ESPERTI E PRINCIPIANTI

Poste Italiane S.p.A Spedizione in A.P. • D.L. 353/2003 (conv.in L.27/02/2004 n.46) art.1 comma 2 DCB ROMA Periodicità mensile • MAGGIO 2006 • ANNO X, N.5 (102)

DATABASE EXPRESS

Usa subito Microsoft SQL Server 2005 e Oracle 10g, i DB di terza generazione

STRUMENTI per creare e gestire gli archivi in modo immediato

CODICE per accedere con Visual Basic.NET e Java

CONSIGLI per migrare da Access a MS SQL

NOVITÀ da conoscere per migliorare il tuo software!

GRATIS ORACLE 10g EXPRESS EDITION NEL CD

.NET
PROBLEMI CON SQL ADDIO!
Trasforma i tuoi dati in classi ed oggetti. Con NHibernate fai tutto con un solo strumento

JAVA
GRAFICA SENZA SEGRETI
Dalle trasparenze alla gestione dei ritagli e dei font, un esempio concreto!

INTERNET
PHP MAILBOT
Progetta un'applicazione Web per la gestione di una Newsletter

.NET ADVANCED
MICROSOFT MESSAGE QUEUE
La guida all'uso del sistema che gestisce lo scambio dei messaggi tra applicazioni

SCAMBIO FILE SU CELLULARI
"TELETRASPORTO" CON BLUETOOTH
Tutto sul protocollo che gestisce il trasferimento dati tra dispositivi mobili

CASI DI STUDIO: JAVA
APPLICAZIONI PRONTE DA ESTENDERE
Realizza un software che interpreta moduli e aggiunge funzioni sviluppate dall'utente

C#
CALCIO & SCOMMESSE
Ecco come funzionano gli algoritmi più efficaci per la riduzione dei sistemi

VISUAL BASIC 6.0
SCRIVI IN VB LEGGI IN WORD
Generiamo dinamicamente documenti RTF dai dati di una nostra applicazione

ALGORITMI DI ORDINAMENTO RADIX SORT, il metodo utilizzato dalle applicazioni Real Time

Anno X - N.ro 5 (102) - Maggio 2006 - Periodicità: Mensile
Reg. Trib. di CS al n.ro 593 del 11 Febbraio 1997
Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo
http://www.ioprogrammo.it

Direttore Editoriale: Massimo Sesti
Direttore Responsabile: Massimo Sesti
Responsabile Editoriale: Gianmarco Bruni
Redazione: Fabio Farnesi
Collaboratori: M. Autiero, C. Bellucci, M. Bigatti, R. Brunetti, L. Buono, L. Caputo, D. De Michelis, C. Durante, F. Grimaldi, M. Scala, G. Sudano

Segreteria di Redazione: Veronica Longo

Realizzazione grafica: Cromatika S.r.l.
Responsabile grafico: Paolo Cristiano
Coordinamento tecnico: Giancarlo Sicilia
Illustrazioni: M. Veltri
Impaginazione elettronica: Elio Monaco

Realizzazione Multimediale: SET S.r.l.
Realizzazione CD-Rom: Paolo Iacona

Pubblicità: Master Advertising s.r.l.
Via C. Correnti, 1 - 20123 Milano
Tel. 02 831212 - Fax 02 83121207
e-mail advertising@edmaster.it
Sales Director: Max Scortegagna
Segreteria Ufficio Vendite: Daisy Zonato

Editore: Edizioni Master S.p.a.
Sede di Milano: Via Ariberto, 24 - 20123 Milano
Tel. 02 831213 - Fax 02 83121330
Sede di Rende: C.da Lecco, zona industriale - 87036 Rende (CS)
Presidente e Amministratore Delegato: Massimo Sesti

ABBONAMENTO E ARRETRATI

ITALIA: Abbonamento Annuale: ioProgrammo (11 numeri) €59,90 sconto 20% sul prezzo di copertina di €75,90 • ioProgrammo con Libro (11 numeri) €75,90 sconto 30% sul prezzo di copertina di €108,90
Offerte valide fino al 30/06/06
Costo arretrati (a copia): il doppio del prezzo di copertina + € 5.32 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 02 831212.
La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 02 83121206, oppure via posta a EDIZIONI MASTER via C. Correnti, 1 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- cc/p n.16821878 o vaglia postale (inviando copia della ricevuta del versamento insieme alla richiesta);
- assegno bancario non trasferibile (da inviarsi in busta chiusa insieme alla richiesta);
- carta di credito, circuito VISA, CARTASI', MASTERCARD/EUROCARD, (inviando la Vs. autorizzazione, il numero della carta, la data di scadenza e la Vs. sottoscrizione insieme alla richiesta).
- bonifico bancario intestato a Edizioni Master S.p.A. c/o Banca Credem S.p.a. c/c 01 000 000 5000 ABI 03032 CAB 80880 CIN Q (inviando copia della distinta insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.

Sostituzioni: qualora nei prodotti fossero rinvenuti difetti o imperfezioni che ne limitassero la fruizione da parte dell'utente, è prevista la sostituzione gratuita, previo invio del materiale difettato.
La sostituzione sarà effettuata se il problema sarà riscontrato e segnalato entro e non oltre 10 giorni dalla data effettiva di acquisto in edicola e nei punti vendita autorizzati, facendo fede il timbro postale di restituzione del materiale.
Inviare il CD-Rom difettoso in busta chiusa a:
Edizioni Master - Servizio Clienti - Via C. Correnti, 1 - 20123 Milano

Assistenza tecnica: ioprogrammo@edmaster.it

Servizio Abbonati:
☎ tel. 02 831212
@ e-mail: servizioabbonati@edmaster.it

Stampa: Arti Grafiche Boccia S.p.a. Via Tiberio Felice, 7 Salerno
Stampa CD-Rom: Neotek S.r.l. - C.da Imperatore - Bisignano (CS)
Distributore esclusivo per l'Italia: Parrini & C S.p.A.
Via Vitorchiano, 81 - Roma

Finito di stampare nel mese di Aprile 2006

Audio Video Foto Bild, A-Team, Calcio & Scommesse, Colombo, Computer Bild Italia, Computer Games Gold, Digital Japan Magazine, Digital Music, Distretto di polizia, DVD Magazine, Filmteca in DVD, Giochi e Programmi per il tuo telefonino, Go!OnLine Internet Magazine, Guide di Win Magazine, Guide Strategiche di Win Magazine giochi, Home Entertainment, Horror mania, I Corsi di Win Magazine, I Fantastici CD-Rom, I film di idea web, I Filmissimi in DVD, I Libri di Quale Computer, I Mitici all'Italiana, Idea Web, InDVD, IoProgrammo, Japan Cartoon, La mia Barca, La mia Videoteca, Le Grandi Guide di ioProgrammo, Linux Magazine, Magnum PI, Miami Vice in DVD, MPC, Nightmare, Office Magazine, Play Generation, Popeye, PC Junior, PC VideoGuide, Quale Computer, Softline Software World, Supercar in dvd, Thriller Mania, Win Junior, Win Magazine Giochi, Win Magazine, Le Collection, Le Femme Fatale del Cinema.

Guerra sui Database

Che l'unico reale elemento di novità nel campo dei linguaggi fosse stato portato da Microsoft con il suo framework 2.0 sembra essere un dato acquisito. Allo stesso modo si lavora già su Windows Vista e di conseguenza su tutti gli argomenti correlati al nuovo sistema nel campo dello sviluppo. Pertanto sembrerebbe che se il mondo degli sviluppatori dovesse interagire unicamente con i linguaggi di sviluppo, ci sarebbe ben poco fermento, almeno riguardo alle applicazioni standalone. Viceversa, in questo periodo la competizione fra le varie software house sembra essersi spostata sui database. Ha cominciao MySQL che con la sua versione 5.0 ha introdotto novità tali che sembrano preannunciare la volon tà di entrare nel segmento aziendale medio alto, ha proseguito PostgreSQL che con l'introduzione di specifici strumenti di installazione in ambiente Windows sembra voler uscire dall'universo Unix dove è già leader. E infine sono scesi in campoi Big. Prima è arrivata Microsoft con il suo MS SQL Server 2005 Express Edition che sembra essere la porta di ingresso proposta ad aziende medio/piccole per approdare a soluzioni più strutturate, ovvero MS SQL Server Standard Edition e a ruota udite, udite si è presentata addirittura Oracle con una sua versione express del server 10g, altrettanto gratuita di quella di Microsoft e con le stesse identiche limitazioni. È facile dedurre che nessuno vuole stare fuori dal mercato dei database. È lì che si è spostata la competizione. E su questo segmento che i grandi giocano una partita importante. Ovviamente tutto va a vantaggio di noi programmatori, che possiamo adesso provare strumenti innovativi a costo completamente gratuito e persino utilizzarli in modo del tutto gratuito e legale nei propri software con, ovviamente le dovute limitazioni. Resta comunque il fatto che siano tutti ottimi strumenti. Noi di ioProgrammo ve li presentiamo tutti, lasciando a voi poi decidere quale sia il migliore per le vostre esigenze. L'eccezionale disponibilità di strumenti ciascuno con le proprie pecularietà ben si adatta alle necessità di personalizzazione che il mercato della programmazione richiede.

Fabio Farnesi ffarnesi@edmaster.it

All'inizio di ogni articolo, troverete un simbolo che indicherà la presenza di codice e/o software allegato, che saranno presenti sia sul CD (nella posizione di sempre \soft\codice\ e \soft\tools\) sia sul Web, all'indirizzo http://cdrom.ioprogrammo.it.

DATABASE EXPRESS

Usa subito Microsoft SQL Server 2005 e Oracle 10g, i DB gratuiti di terza generazione

- ✔ Scaricali dalla rete e installali sul tuo PC senza costi di licenza
- ✔ Utilizzali con Visual Basic.NET e Java
- ✔ Migra da Access a MS SQL utilizzando il formato mdf invece del vecchio MDB
- ✔ Scopri le espressioni regolari, i lob e le funzionalità avanzate di Oracle

pag. 16

ORACLE 10g XE: NUOVO E GRATIS

Oracle introduce una nuova versione del suo database con nuove funzionalità per velocizzare e ottimizzare lo sviluppo. E rende il tutto gratis, per ogni applicazione pag. 24

QUALCHE CONSIGLIO UTILE

I nostri articoli si sforzano di essere comprensibili a tutti coloro che ci seguono. Nel caso in cui abbiate difficoltà nel comprendere esattamente il senso di una spiegazione tecnica, è utile aprire il codice allegato all'articolo e seguire passo passo quanto viene spiegato tenendo d'occhio l'intero progetto. Spesso per questioni di spazio non possiamo inserire il codice nella sua interezza nel corpo dell'articolo. Ci limitiamo a inserire le parti necessarie alla stretta comprensione della tecnica.

http://forum.ioprogrammo.it

RIVISTA + 2 CD-ROM in edicola

Prodotti del mese

Oracle 10g Express Edition

Per la prima volta in versione gratuita, arriva il DB per le aziende

Oracle si è sempre caratterizzata per la sua volontà di servire il mercato B2B, ed è ancora così. Per anni il suo database è stato il leader nel segmento aziendale. Fino a ieri nessuna o quasi nessuna applicazione destinata alle aziende medio piccole ha potuto usufruire della potenza di questo DB soprattutto in relazione agli enormi costi da sostenere. Oggi tutto è cambiato e Oracle per la prima volta presenta una versione Express gratuita del proprio strumento principe. Si tratta di una novità non da poco. Questa versione è identica in tutto e per tutto alla versione standard eccetto che per il fatto di poter servire un solo processore, gestire fino a un massimo di 4GB di dati, e utilizzare un solo GB di RAM. I linguaggi con cui può essere integrato tramite appositi connector sono i più svariati.

[pag.106]

VS.PHP For Visual Studio 2005

L'Addin per usare Visual Studio come editor per PHP

Si tratta di una vera e propria novità. Utilizzare uno degli editor più evoluti al mondo per programmare per il web utilizzando il linguaggio PHP. Alcuni potrebbero pensare che si tratti in fondo del solito editor con sintassi colorata, e invece tutte le funzioni avanzata ci sono tutte, dalla syntax highlighting alla code complexion, più alcune funzionalità decisamente avanzate che non si trovano tipicamente nemmeno in editor nativi per PHP come ad esempio il supporto a smarty il template designer di PHP utilissimo per separare la logica di programmazione da quella di design. Si tratta insomma di uno strumento di eccezionale rilevanza che può diventare realmente significativo per coloro che fino ad ora non hanno trovato un IDE sufficientemente evoluto per lo sviluppo delle proprie Web Application.

[pag.107]

AXIS 1.3

Il framework per lo sviluppo di Web Service in Java

Che i Web Service rappresentino una risorsa enorme per la programmazione è ormai noto, in questo stesso numero di ioProgrammo diamo enorme spazio all'argomento proprio con una serie di esempi pratici per lo sviluppo. Mancano dai nostri esempi proprio quelli relativi a Java di cui ci occuperemo nei numeri successivi della rivista. Axis è comunque il tool fondamentale per poter sviluppare WS con il linguaggio di SUN. Il suo utilizzo è abbastanza semplice, e supporta tutte le caratteristiche avanzate come ad esempio la generazione dinamica dei WSDL oppure il debugging delle trasmissioni SOAP Lavora in congiunzione stretta con Tomcat e il suo deployment si riduce a copiare i file nella directory corretta.. Uno strumento indispensabile di cui non mancheremo di parlare ulteriormente.

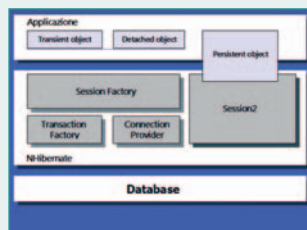
[pag.106]

NHIBERNATE 1.0.2

Il tool di persistenza dei dati per eccellenza

Ce ne parla Giancarlo Sudano in questo stesso numero di ioProgrammo. Nhibernate è il porting del famoso Hibernate per Java su piattaforma .NET.

La sua utilità è estrema, consente infatti di trasformare le normali Query SQL in oggetti e classi di modo che siano omogenei con il codice a cui uno sviluppatore è abituato. Oltre a questo Nhibernate è un tool di persistenza che consente di "salvare" lo stato dei dati su uno storage, anche questo in modo indipendente dal linguaggio SQL. A fare da ponte fra l'architettura del database e il linguaggio ad oggetti sarà un semplice file XML. La coomodità di uno strumento del genere é innegabile, tanto che la maggior parte dei linguaggi si sta orientato a dotarsi internamente di questo genere di tecnologia.

[pag.107]

I contenuti del libro

Imparare JAVA

Quando qualche anno fa, quando Java ha fatto il suo ingresso sul mercato assomigliava a qualcosa di utile per inserire un miniprogramma all'interno di una pagina Web, era l'era delle Applet. A distanza di quasi 10 anni dal suo primo rilascio, il linguaggio si è evoluto in modo sostanziale, fino a diventare il punto di riferimento per le applicazioni aziendali come dell'OpenSource e persino uno dei linguaggi più didattici per il mondo universitario. Ivan Venuti ce ne illustra passo dopo passo le varie caratteristiche. Si parte dall'ABC dei costrutti elementari fino ad arrivare a trattare argomenti avanzati che coinvolgono l'uso dei package e l'ottimizzazione del software. Un manuale di riferimento pratico ed efficiente che non deve mancare all'utente esperto come a chi inizia.

IL MANUALE RAPIDO PER PROGRAMMARE SOFTWARE PER WINDOWS E PER LINUX CON UN UNICO LINGUAGGIO

- I fondamenti della tecnologia
- La programmazione orientata agli oggetti
- I package e l'organizzazione delle classi
- Ottimizzazione del software

GLI ALLEGATI DI IOPROGRAMMO

▼ Programmare in Team

C'era una volta il singolo programmatore, che da solo forte delle sue conoscenze riusciva a soddisfare le "poche esigenze" dei clienti. Nel corso degli anni la domanda è cambiata. Le esigenze sono diventate del tutto diverse e sempre più complesse. Difficilmente un singolo programmatore sarebbe in grado di produrre un'applicazione completa in grado di servire un'azienda medio grande. Per questo è nata la programmazione in team che consente di affidare singoli moduli, piccole entità ad un singolo per poi raggruppare tutto sotto un unico cappello. Questo ovviamente ha introdotto nuovi problemi. Chi si occuperà di progettare l'intera applicazione? di quali strumenti avrà bisogno? chi si occuperà di sincronizzare le varie revisioni? la risposta di Microsoft a tutto questo e a molto altro è contenuta in Visual Studio Team System, ovvero lo strumento pensato da MS per consentire a piccoli e grandi gruppi di lavoro di lavorare insieme in modo omogeneo. Si tratta di uno strumento molto complesso che tuttavia implicitamente contiene tutte le soluzioni per rendere immediatamente produttiva qualunque software house.

I VIDEOCORSI PER PROGRAMMARE BENE

4 WEBCAST UFFICIALI MICROSOFT

IN ESCLUSIVA GRATIS NEL CD "I CORSI DI FORMAZIONE" DA SEGUIRE COMODAMENTE SUL PC

msdn WEBCAST

VISUAL STUDIO TEAM SYSTEM

- Introduzione e logica
- Tecniche per l'application Modeling
- Gestione del codice sorgente
- Cosa è e come funziona il project Management

INFORMAZIONI SU MSDN WEBCAST
http://www.microsoft.it/msdn/webcast_msdn
http://forum.ioprogrammo.it

FAQ

Cosa sono i Webcast MSDN?

MSDN propone agli sviluppatori una serie di eventi gratuiti online e interattivi che approfondiscono le principali tematiche relative allo sviluppo di applicazioni su tecnologia Microsoft. Questa serie di "corsi" sono noti con il nome di Webcast MSDN

Come è composto tipicamente un Webcast?

Normalmente vengono illustrate una serie di Slide commentate da un relatore. A supporto di queste presentazioni vengono inserite delle Demo in presa diretta che mostrano dal vivo come usare gli strumenti oggetto del Webcast

Come mai trovo riferimenti a chat o a strumenti che non ho disponibili nei WebCast allegati alla rivista?

La natura dei WebCast è quella di essere seguiti OnLine in tempo reale. Durante queste presentazioni in diretta vengono utilizzati strumenti molto simili a quelli della formazione a distanza. In questa ottica è possibile porre domande in presa diretta al relatore oppure partecipare a sondaggi etc. I WebCast riprodotti nel CD di ioProgrammo, pur non perdendo nessun contenuto informativo, per la natura asincrona del supporto non possono godere dell'interazione diretta con il relatore.

Come mai trovo i WebCast su ioProgrammo

Come sempre ioProgrammo cerca di fornire un servizio ai programmatori italiani. Abbiamo pensato che poter usufruire dei WebCast MSDN direttamente da CD rappresentasse un ottimo modo di formarsi comodamente a casa e nei tempi desiderati. Lo scopo tanto di ioProgrammo, quanto di Microsoft è infatti quello di supportare la comunità dei programmatori italiani con tutti gli strumenti possibili.

Su ioProgrammo troverò tutti WebCast di Microsoft?

Ne troverai sicuramente una buona parte, tuttavia per loro natura i webcast di Microsoft vengono diffusi anche OnLine e possono essere seguiti previa iscrizione. L'indirizzo per saperne di più è: http://www.microsoft.it/msdn/webcast/msdn segnalo nei tuoi bookmark. Non puoi mancare.

L'iniziativa sarà ripetuta sui prossimi numeri?

Sicuramente si.

News

ENNESIMO EXPLOIT PER EXPLORER

Non c'è pace per il browser di Microsoft. Questa volta è una falla nel metodo CreateTextRange() presente sia in IE6 che, udite udite, nella beta 2 di IE7.
L'exploit proviene direttamente dal notissimo gruppo di hacker *unl0ck .net*, tuttavia sarebbe stato modificato da moltissime altre mani tanto che in molti pensano che a breve si trasformerà in un nuovo potente virus. Al momento in cui scriviamo Microsoft non ha rilasciato alcuna patch e consiglia di disattivare JavaScript, tuttavia ha già dichiarato che il problema sarà risolto con la pubblicazione dei bollettini del mese di Aprile

ANCORA RIMANDATO IL LANCIO DI OFFICE 2007

Il rilascio della nuova versione della suite per l'ufficio di Microsoft dovrà essere allineato al rilascio di Windows Vista. Questa è la decisione del gigante di Redmond su uno dei suoi software di punta. In particolare poiché Windows Vista è atteso per l'inizio del 2007 allo stesso modo Office 2007 vedrà la luce nei primi mesi del nuovo anno. Non si tratta di una novità di poco conto se si tiene conto che il mercato dei PC attende l'arrivo del nuovo sistema come una sorta di ancora di salvezza che possa in una qualche maniera incentivare all'acquisto di nuovo hardware.
D'altra parte da anni questo mercato sopravvive con alterna fortuna, ma senza mai raggiungere i picchi di gloria che aveva vissuto dai tempi del 386 fino ai primi pentium. Inoltre Microsoft sta conducendo una campagna molto intensa su Windows Vista e sui prodotti ad esso correlati. È più che giusto che desideri rilasciare il sistema senza farsi stressare dai tempi di uscita, quanto piuttosto osservando i dettami della qualità. Un nuovo sistema colmo di bug non aiuterebbe nessuno.

PRIME INDISCREZIONI SU PHP 6

L'arrivo di PHP 5 era stato salutato come un'eccezionale novità. Ed in effetti lo è stato! Un modello ad oggetti completamente rinnovato metteva in mano al programmatore uno strumento potente e ben curato per compiere il necessario salto di qualità dalla classica programmazione procedurale ad una più moderna e funzionale programmazione ad oggetti. E tuttavia PHP è forse l'unico linguaggio di successo al mondo dove la programmazione ad oggetti ha ottenuto scarso o poco successo. Di fatto la maggior parte dei programmatori sono rimasti attaccati alla buona vecchia programmazione procedurale del PHP 4. Pochissime fra le web application più note sono state convertite alla nuova versione. In questo scenario che vede ancora PHP 4 fare da padrone e PHP 5 stentare a decollare si inseriscono le prime indiscrezioni sul futuro PHP 6. Le novità riguarderebbero un migliorato supporto ad Unicode, che consentirebbe di sviluppare applicazioni non curandosi della tipologia di carattere da usare ma lasciando a PHP il compito di scegliere il formato di codifica per l'input e per l'output. Alcune modifiche "minori" riguarderebbero determinate direttive dell'interprete che fino ad ora avevano creato più problemi che benefici. In particolare scomparirebbero le tanto contestate register_ globals, magic_quotes e safe_mode. Una novità apparentemente poco entusiasmante riguarderebbe l'uso di APC: "Alternative PHP Cache". Fino ad oggi in pochi avevano sfruttato questa estensione di PHP. APC consente di "compilare" degli script PHP e mantenerli in una cache di modo da ottimizzare le performance di un sito che ne fa uso, nella nuova

GERMANIA: RISCHIA DUE ANNI CHI SCARICA!

È il frutto della legge appena approvata dal parlamento federale tedesco. Nessuna indulgenza per i pirati! Chi scarica illegalmente contenuti protetti da copyright è in tutto e per tutto assimilabile a chi compie un furto, per tale motivo la condanna prevista è di 2 anni di reclusione. Si tratta di una delle leggi fino ad ora più dure in Europa. La Germania sceglie di adottare il pugno di ferro contro la pirateria che, secondo i politici e gli economisti tedesch.i starebbe gettando in ginocchio il mercato della musica e della cinematografia. Naturalmente sono state dure le reazioni delle organizzazioni dei consumatori che fanno notare come il fenomeno abbia raggiunto proporzioni tali da meritare un'attenzione diversa. Stando così le cose, viceversa, una famiglia tedesca potrebbe vedersi recapitare un mandato di cattura a causa delle scorribande informatiche di un figlio sedicenne. Probabilmente si tratta di un metodo per debellare un fenomeno che realmente mette a rischio un mercato che è una fonte importante per l'economia tedesca ed europea in genere. E' anche vero che mentre la tecnologia ha fatto passi da gigante nel campo delle comunicazioni, nessun tentativo da parte delle major è stato fatto per adeguarsi ai nuovi tempi. E' la solita storia che ormai ci portiamo avanti da qualche anno. Si adegueranno le major alle nuove tecnologie, oppure le carceri tedesche saranno sovraffolate di sedicenni amanti della buona musica?

versione di PHP questa estensione sarà inclusa nel core del linguaggio. Una novità decisamente più eclatante riguarderebbe il già tanto discusso modello ad oggetti, di fatto verrebbe introdotto il tanto caro concetto di "namespace" , che consente di raggruppare sotto un unico "ombrello", variabili, oggetti e classi. Infine grandi movimenti coinvolgerebbero le estensioni tipicamente disponibili per il linguaggio. Alcune storiche estensioni come "ereg" sparirebbero a favore di altre più moderne come "PCRE". Una lista piuttosto completa di cosa accadrà a PHP con la nuova versione è disponibile all'indirizzo: http://php.net/~derick/meeting-notes.html.

565 MILIONI PER LA TECNOLOGIA

È quanto stanziato dal ministero per l'innovazione tecnologica per la creazione di 36 distretti digitali, piccole cittadelle che svolgeranno funzione di polo d'aggregazione per lo sviluppo dell'high tech in Italia. In particolare saranno cinque le priorità su cui il governo intende concentrare i propri sforzi per abbattere la barriera digitale che divide il sud dal nord del paese. "Sviluppo di azioni per la promozione di una cultura dell'innovazione digitale", ovvero dall'e-learning alla formazione finalizzata alla riqualificazione delle risorse umane - "sviluppo delle infrastrutture immateriali", per agevolare la domanda di connetività, le reti wireless e la produzione di applicazioni innovative nel campo della produzione logistica e del marketing - "applicazione del codice dell'amministrazione digitale", ovvero si intende dare effettiva concretezza alle norme già vigenti sull'informatizzazione della pubblica amministrazione e sui rapporti con il cittadino, la dove la P.A. non sempre è stata capace di adottare le nuove tecnologie come una risorsa concreta per fornire servizi efficienti - "Sviluppo di piattaforme tecnologiche a supporto dei processi produttivi per nuovi distretti" , per sostenere aziende o consorzi che agiscano come "consulenti d'impresa" dei distretti tecnologici – infine "Finanza per l'innovazione", ovvero un fondo per la creazione di imprese innovative.
Si tratta di un finanziamento importante a cui il ministro Stanca affida la creazione di un ecosistema digitale fatto da piccole, grandi imprese e università che tutte insieme mettono in moto un meccanismo integrato per il rilancio del mercato informatico in Italia.

FESTA DEGLI SCONOSCIUTI AL JOLT AWARD 2006

Il Jolt Award è uno dei riconoscimenti più prestigiosi che ogni anno vengono riconosciuti ai programmi che si sono distinti nell'ingegneria del software. La manifestazione è talmente prestigiosa da potersi permettere di non essere minimamente influenzata dai grandi nomi che tipicamente affollano il panorama informatico. Così quest'anno a farla da padrone nel campo dei Web Development Tools è addirittura lo sconosciutissimo ai più Rails 1.0, nato dall'ancora meno noto Ruby. Mentre nel campo dei Design Tools si afferma l'ignoto Lattix. Certo non mancano i nomi noti, a vincere il premio nella categoria Database è il nuovo nato Microsoft Sql Server 2005 che si lascia dietro le spalle il Berkeley DB 4.4, solo al quarto posto MySQL 5.0. La situazione si ribalta nel premio riservato all'enterprise project management dove ad avere la meglio è stato Welcom Risk 2.6 seguito da Corticon Business Roles Management e Jboss, si piazza solo al quarto posto Microsoft Visual Studio Team System 2005 che invece domina nel campo degli ambienti di sviluppo lasciandosi dietro le spalle Eclipse 3.1, intelliJ Idea 5.0 e Komodo 3.5. Nel campo dei framework prevale ancora Microsoft con il suo .NET 2.0 a seguire Dundas Chart for .NET, QT 4.0 e solo quarto lo Spring Framework di Java. Vincitore su tutti con grande merito e prestigio Visual Studio Professional Edition che vince nella categoria "Hall of Fame" senza nessun rivale, decisamente un successo. È importante notare come nel campo dell'alta tecnologia le sfide siano poco influenzate dalle politiche di marketing, mentre invece a vincere è sempre la buona progettazione e l'utilità per i consumatori.

550.000 DOMINI FATTI FUORI DA UN DDOS

In Italia sicuramente non è uno dei register più famosi ma lo è sicuramente nel mondo, si tratta di Joker.com. L'azienda sarebbe stata fatta oggetto di un violento attacco di DdoS tale che circa 550.000 domini sarebbero stati messi fuori gioco. Il DdoS è purtroppo una pratica difficile da intercettare, anche se non sono ancora note le modalità con cui sia stato possibile mettere in corto circuito un'azienda del calibro di Joker.com con un attacco di tipo DdoS. Al momento si pensa a tecniche di tipo ricorsivo ai DNS.

ARRIVA L'ECLIPSE PHP INTEGRATED DEVELOPMENT ENVIRONMENT

Così dopo avere a lungo frequentato gli ambienti periferici di Eclipse, il PHP IDE ha finalmente raggiunto la maturità necessaria per diventare a pieno titolo uno dei progetti di maggiore rilievo all'interno della piattaforma di sviluppo. Al momento saranno tre gli aspetti su cui verrà focalizzata dagli sviluppatori. Prima di tutto il PHP IDE Core che conterrà le specifiche di linguaggio il code builder e il formatter, oltre che tutta una serie di strutture necessarie a far ben funzionare l'intero ambiente. PHP IDE UI è la sezione che si occuperà di definire l'interfaccia utente, infine il PHP IDE DEBUG ovvero la sezione dedicata all'implementazione di un debugger finalizzato a rendere la vita più semplice al programmatore. Così finalmente al classico ZEND Studio si affiancherà un IDE maturo e potente con alle spalle un team di sviluppatori capace di produrre un IDE significativo per i programmatori PHP. Vedremo se il nascituro PHP IDE basato su Eclipse sarà all'altezza delle ben note straordinarie capacità di Zend oppure se annegherà anch'esso nella proverbiale fame di risorse tipica di Eclipse.

LA CINA CONTRO LA PIRATERIA

Sarebbero 14 le fabbriche costrette alla chiusura da un blitz compiuto dal governo e teso a combattere la pirateria. Le strutture in questione sarebbero state dotate di attrezzature tali da poter produrre quantitativi enormi di materiale contraffatto. Che la Cina nota per essere la patria dell'imitazione sia scesa in campo contro il fenomeno della duplicazione illegale è sintomo di quanto i nostri tempi stiano cambiando. Certo c'è differenza fra imitazione e duplicazione esatta, tuttavia certamente fino ad ora non erano mancati sul mercato i prodotti oggetto di contraffazione provenienti dai mercati asiatici.

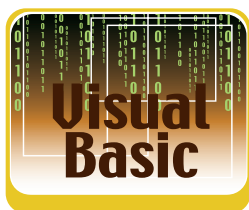
MICROSOFT ABBRACCIA L'OPENDOCUMENT

Nel nome dell'interoperabilità era nata non molto tempo fa l'ODF Alliance che vedeva la partecipazione di oltre 35 membri tra cui IBM, Sun, Novell, e Red Hat oltre che molti enti pubblici come l'American Library Association. Scopo dell'ODF Alliance è stendere le linee guida per un formato OpenSource per la descrizione dei documenti. La spinta alla formazione del consorzio era fornita dalla preoccupazione degli stati nel dover utilizzare un formato proprietario che li vincolasse all'adozione di un solo sistema operativo e comunque li soggiogasse agli intenti di un'azienda privata.
Viceversa l'adozione di un formato OpenSource condiviso da larga parte delle applicazioni garantirebbe la necessaria libertà di movimento ai governi.
Recentemente al gruppo delle aziende costituenti l'ODF si è aggiunta proprio Microsoft. L'ingresso del gigante di Redmond, se da un lato apre nuovi scenari, dall'altro nelle menti dei più maliziosi lascia insorgere il sospetto che la software house di Bill Gates voglia in qualche modo rallentare l'adozione del nuovo formato a favore del proprio standard Open XML.
Naturalmente Microsoft ha respinto totalmente le accuse.

GLI UTENTI AIUTANO MICROSOFT

Per la prima volta Microsoft utilizza un sistema largamente diffuso fra il software OpenSource ovvero quello di un sito pubblico destinato agli utenti che vogliano segnalare eventuali bachi riscontrati nell'uso dei programmi di MS. Il sito è raggiungibile all'indirizzo https://connect.microsoft.com/default.aspx. Nella home page del progetto si legge "Collaborazione attiva allo sviluppo dei prodotti Microsoft" e ancora "Il sito consente di entrare in contatto con gli sviluppatori, i responsabili dei prodotti e gli altri membri dei team di sviluppo allo scopo di contribuire al costante e decisivo miglioramento dei prodotti." Partecipare è semplice, basta accedere con il proprio passport login , cliccare su "Invia Commento" e segnalare il bug scovato utilizzando uno dei programmi di MS. Il bug in questione se ritenuto valido sarà preso in gestione da uno sviluppatore MS e la risposta sarà pubblica. Il sistema è molto simile a quello utilizzato dal noto bugzilla che tipicamente accompagna lo sviluppo di qualunque software OpenSource.

SQL SERVER 2005 LEGGERO E FLESSIBILE

IL DATABASE DI MICROSOFT SI RINNOVA. FRA LE GRANDI NOVITÀ: I DATI VENGONO CONSERVATI IN FILE MDF CON UNA LOGICA SIMILE A QUELLA DI ACCESS E UNA VERSIONE DEL TUTTO GRATUITA...COS'ALTRO ANCORA?

Gli ultimi mesi del 2005 hanno visto nascere la famiglia di prodotti .NET 2005 della Microsoft. In questo articolo ci occuperemo di introdurre SQL Server 2005 e del suo utilizzo con Visual Basic .Net 2003. La famiglia di prodotti SQL Server 2005 prevede quattro nuove edizioni:
Express, Workgroup, Standard ed Enterprise.
La versione Express, è una versione free che permette di gestire database fino a 4GB, e con essa sono forniti, sempre gratis, un Management Tool e un Report Tool.
Le altre tre versioni a pagamento, permettono di gestire database senza limiti di dimensione e forniscono lo strumento Management studio che rappresenta una piattaforma completa di gestione per SQL Server.
Trovate maggiori informazioni sulle differenze fra la versione Express e le altre nel box presente in questo stesso articolo.

SQL NATIVE CLIENT

SQL Native Client è la nuova tecnologia di accesso ai dati installata da SQL Server 2005. Essa racchiude in un'unica libreria, le caratteristiche del provider SQL OLE DB del driver ODBC e nuove funzionalità. Poiché SQL Native Client non dipende esplicitamente da una particolare versione di MDAC (Microsoft Data Access Components) è ancora possibile utilizzare la versione di MDAC installata sul nostro computer. Quindi si può accedere ad SQL Server 2005, utilizzando direttamente le vecchie tecnologie di accesso, che però non permettono di utilizzare le nuove features quali XML data type, gli UDT (tipi definiti dall'utente) i multiple active resultsets (MARS) ecc.
Un'altra feature introdotta con SQL Native Client è legata al trasporto dei database, con SQL 2005 tutte le informazioni del database possono essere racchiuse e trasportate nel

REQUISITI DI SISTEMA PER SQL SERVER 2005

PROCESSORE
Velocità 500 MHz o superiore (consigliato a 1 GHz o superiore)

SISTEMI OPERATIVI
***SQL Server 2005 Enterprise Edition* e *Standard Edition* possono essere eseguiti sui seguenti sistemi operativi:**

- **Windows Server 2003 Standard Edition**
- **Windows Server 2003 Enterprise Edition**
- **Windows Server 2003 Datacenter Edition**
- **Windows Small Business Server 2003 Standard Edition**
- **Windows Small Business Server 2003 Premium Edition**
- **Windows 2000 Server**
- **Windows 2000 Advanced Server**
- **Windows 2000 Datacenter Server**

***SQL Server 2005 Standard Edition* può essere eseguito anche su Windows XP Professional Service Pack 2 o successivo.**

***SQL Server 2005 Evaluation Edition* e *Workgroup Edition* possono essere eseguiti anche su Windows 2000 Professional.**

SQL Server 2005 Developer Edition* ed *Express Edition* possono essere eseguiti sui sistemi operativi indicati sopra e in più sui seguenti sistemi operativi: *Windows XP Home Edition* e *Windows Server 2003 Web Edition.

MEMORIA
***Express Edition:* 128 MB (consigliata da 512 MB in su). Tutte le altre versioni 512 MB (consigliata da 1 GB in su)**

SPAZIO SU DISCO
350 MB di spazio libero su disco rigido per l'installazione completa. Per installare i database di esempio sono necessari 390 MB.

file MDF. Questo, sicuramente, rappresenta una rivoluzione per gli sviluppatori di Web Applications che potranno mantenere una copia del file MDF in una directory del sito ed utilizzarla come un file Microsoft Access. Naturalmente anche in questi scenari SQL Server deve essere sempre disponibile.

SQL SERVER MANAGEMENT STUDIO

Le principali funzionalità dell'ambiente di sviluppo di SQL Server 2005 sono accessibili mediante Management Studio.
Potremmo definire Management Studio come la centrale di controllo di Microsoft SQL Server, esso offre tutti gli strumenti per eseguire e monitorare le funzioni vitali di SQL Server. Management Studio combina in un unico ambiente le funzionalità che nelle precedenti versioni di SQL Server erano offerte da Enterprise Manager, Query Analyzer e Analysis Manager. In questo modo, gli amministratori di database dispongono di un'unica utilità completa ed integrata, che combina strumenti grafici di facile utilizzo.
Per avviare SQL Server Management Studio, si deve cliccare sul pulsante Start della barra delle applicazioni, scegliere Tutti i programmi Microsoft SQL Server 2005 e quindi SQL Server Management Studio.

IL COMPONENTE ESPLORA OGGETTI

Il componente principale di SQL Server Management Studio è il componente Esplora oggetti.
Esplora oggetti è uno strumento integrato che presenta un'interfaccia utente in grado di visualizzare e gestire gli oggetti in qualsiasi tipo di server.
La struttura di *Esplora oggetti* è quella classica in cui le informazioni sono raggruppate in cartelle. Per espandere le cartelle, in modo da visualizzare informazioni più dettagliate, si deve cliccare sul segno più (+) o fare doppio clic sulla cartella.
La prima volta che si espande una cartella,

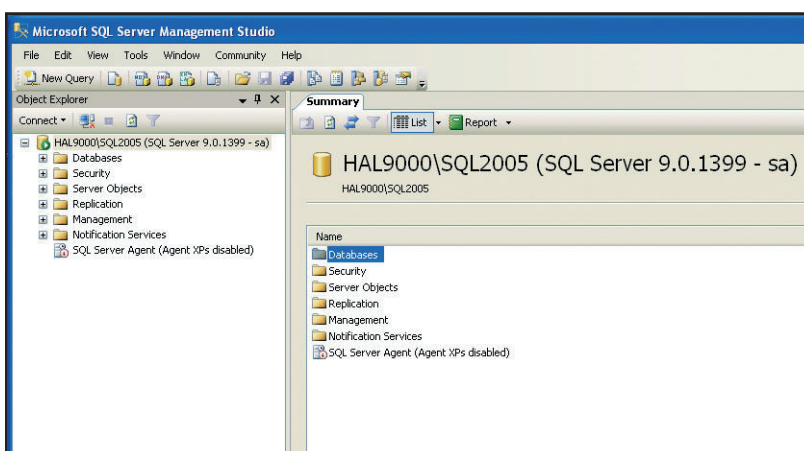
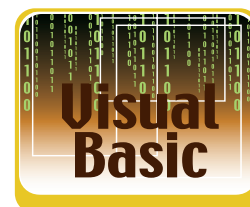
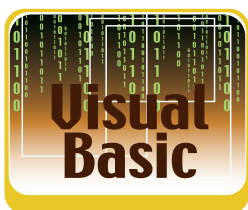



Fig. 1: *La GUI di Management Studio*

LE PRINCIPALI NOVITÀ DI SQL SERVER 2005

**Sql Server 2005, rappresenta senza dubbio un elemento di rottura rispetto alle precedenti versioni e per più di un motivo. Prima di tutto ne esiste una versione *"Express"* completamente gratuita scaricabile all'indirizzo http://msdn.microsoft.com/vstudio/express/sql/default.aspx. La versione *Express* è identica in tutto e per tutto alle altre edizioni di SQL server. Le limitazioni riguardano: il numero di CPU utilizzabili, in SQL Server Express è possibile utilizzare una sola CPU, la quantità massima di memoria utilizzabile limitata ad 1 GB per la versione *Express,* l'assenza di tool aggiuntivi quali: *"Analysis Services, Reporting Services, Data Transformation Services, Notification Services".* Nonostante questo, SQL Server Express rappresenta un ottimo punto di ingresso per chi vuole provare a sviluppare applicazioni che facciamo uso di Database in tecnologia Microsoft. In caso di utilizzi avanzati si potrà comunque upgradare a versioni più evolute senza troppi problemi. Rispetto alle versioni precedenti SQL Server 2005 si caratterizza per diventare una completa piattaforma di sviluppo. In particolare le funzionalità più interessanti sembrano essere la presenza di un *"Service Broker"* all'interno dell'engine, che consente di aggirare i limiti dell'assenza di connettività in situazioni di ambienti distribuiti. In sostanza il *Service Broker* gestisce una coda interna di richieste che verranno eseguite non appena le condizioni lo consentiranno, ad esempio al ritorno della piena connettivitità o quando il server sarà abbastanza scarico per effettuare le operazioni in modo corretto. Ovviamente in assenza di connessione non sarà possibile effettuare nessuna richiesta a un server esposto in rete, questo problema si aggira installando un SQL Server locale alla macchina, effettuando la richiesta al Server locale che gestirà la coda, ed al ritorno della connessione provvederà ad aggiornare il server remoto. I costi sono ovviamente nulli essendo il server locale in versione *Express.* Un'ulteriore importante innovazione è quella relativa ai tipi di dati, accanto al tradizionale *T-SQL* è ora possibile scrivere delle query molto complesse utilizzando direttamente il *Framework .NET.* Infine c'é un'altra integrazione con XML, adesso maneggiato in modo nativo e quindi facilmente accessibile e manutenibile. Ulteriori novità sono rappresentate dagli *Integration Services* ovvero un sistema di integrazione che consente tramite un linguaggio batch di sviluppare procedure ad Hoc per la conversione di sistemi eterogenei. In questo modo è per esempio possibile ricevere elementi da un *ocr*, piuttosto che da un file *.csv* per poi farli confluire tramite *IS* in un unico sistema. Della possibilità di salvare i dati in file esterni con estensione *.mdf* estremamente portabili con la logica dei vecchi file *mdb,* parleremo direttamente nel corso dell'articolo.
Altri elementi che non citiamo sono quelli di reportistica e analisi di cui ci occuperemo successivamente.**

Esplora oggetti chiederà al server le informazioni per popolare la struttura, nel frattempo è possibile eseguire altre funzioni.
L'elenco del contenuto delle cartelle non viene aggiornato automaticamente in modo da risparmiare risorse, nel caso in cui vi siano molti oggetti da visualizzare.
Per aggiornare l'elenco degli oggetti all'interno di una cartella, cliccare con il pulsante destro del mouse sulla cartella e quindi scegliere *Refresh*. Quando *Esplora oggetti* è connesso a un server è possibile aprire una nuova finestra *Editor del codice*. Per aprire la finestra di *Editor del codice*, si deve cliccare con il pulsante destro del mouse sul nome del server in *Esplora oggetti* e quindi scegliere *New query*. La finestra delle query è solitamente divisa in tre riquadri. Il riquadro a sinistra consente di esaminare oggetti disponibili sull'istanza del server del database selezionato. Si deve utilizzare il riquadro in alto a destra per immettere le query. Il riquadro in basso a destra mostra i risultati. Dalla barra degli strumenti si può selezionare un database tra quelli disponibili.

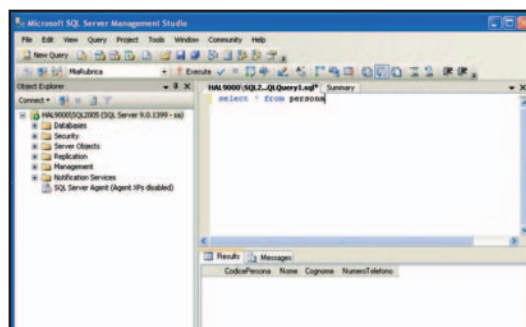



Fig. 2: L'editor del codice

I NUOVI STRUMENTI

Gli strumenti ora incorporati in *SQL Server Management Studio* sono:

EDITOR DEL CODICE
Un editor completo per la scrittura e la modifica degli script che sostituisce *Query Analyzer,* incluso nelle versioni precedenti di SQL Server. *SQL Server Management Studio* include quattro versioni di *Editor* del codice: l'editor di query SQL, l'editor di query MDX, l'editor di query XMLA e l'editor di query *SQL Mobile.*

ESPLORA OGGETTI
Per l'individuazione, la modifica, la creazione di script o l'esecuzione di oggetti appartenenti a istanze di SQL Server.

ESPLORA MODELLI
Per l'individuazione e la creazione di script di modelli.

ESPLORA SOLUZIONI
Per l'organizzazione e l'archiviazione di script correlati come parti di un progetto.

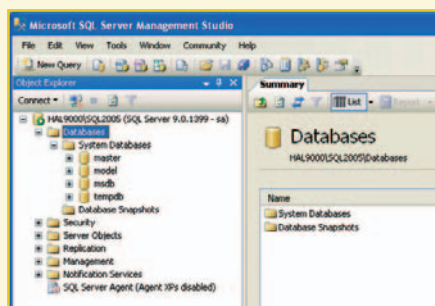
FINESTRA PROPRIETÀ
Per la visualizzazione delle proprietà correnti degli oggetti selezionati.

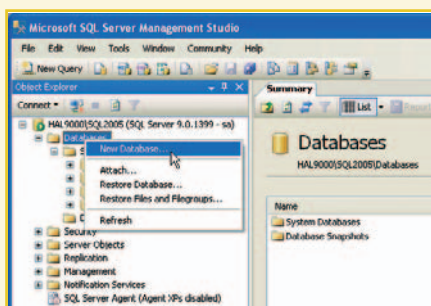
ACCEDERE AL DATABASE DA VISUAL BASIC .NET

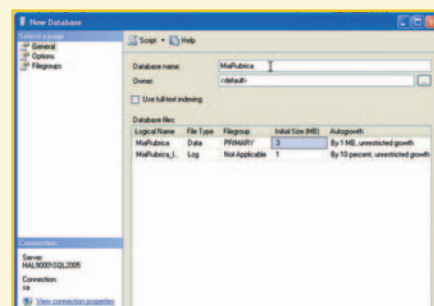
La prima azione da eseguire quando si vuole utilizzare una fonte dati è quella di aprire una connessione verso quest'ultima, ciò significa creare un oggetto SqlConnection, impostare la stringa di connessione, tramite la proprietà ConnectionString, ed aprire la

CREAZIONE DI UN DATABASE SQL SERVER 2005

Ecco i passi necessari per creare un nuovo database in SQL Server 2005

> INIZIAMO

1 La prima operazione da compiere è quella di espandere il ramo del Server fino alla cartella *Databases.*
Otterremo l'elenco dei Database già creati nell'ambiente.

> UN NUOVO DB

2 Clicchiamo con il tasto destro del mouse sulla cartella *Databases*, e dal menu contestuale selezioniamo la voce: *New Database.* In questo modo si aprirà la finestra di dialogo *New Database*.

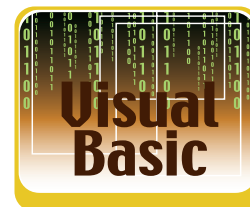
> QUALE NOME?

3 Nella finestra di dialogo *New Database* è possibile selezionare, dall'albero di sinistra, tre differenti sezioni. Nella sezione *General* dobbiamo indicare il nome logico del nuovo db ad esempio: *MiaRubrica.*

connessione tramite il metodo Open.
Per operare in modalità connessa, questi punti devono essere eseguiti soltanto alla partenza dell'applicazione, mentre alla fine dell'applicazione si deve chiudere la connessione tramite il metodo Close. Per operare in modalità disconnessa, invece, i punti precedenti devono essere eseguiti ogni volta che si deve accedere al database. In pratica, ogni volta che si deve accedere ad un database si devono scrivere le seguenti istruzioni:

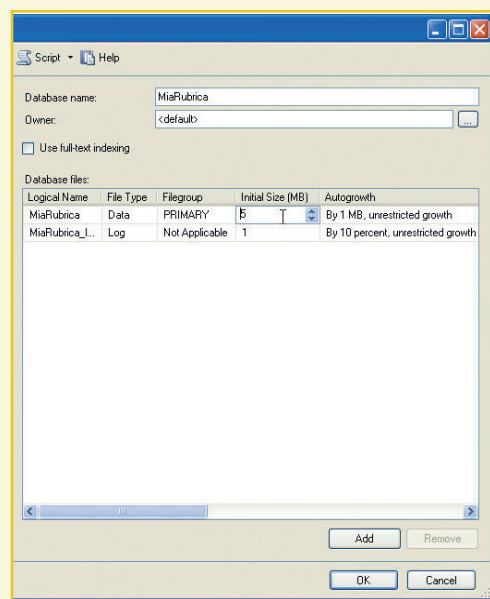
```
Dim ObjConnection As New SqlConnection()
ObjConnection.ConnectionString = "Initial
                          Catalog=MiaRubrica;Data
 Source=localhost;Integrated Security=SSPI;"
ObjConnection.Open()
'operazioni su database
ObjConnection.Close()
```

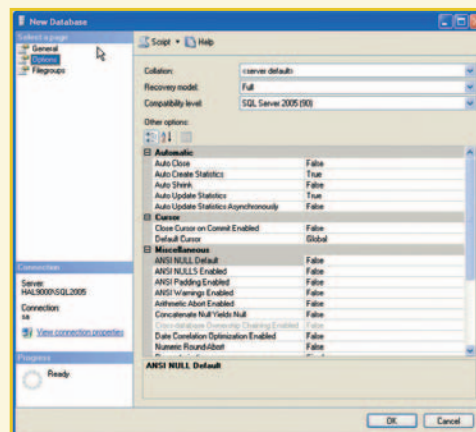
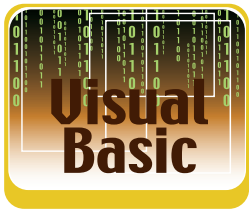
L'OGGETTO SqlCommand

Per interagire con un database, VB.Net mette a disposizione l'oggetto SqlCommand.
L'oggetto SqlCommand deve essere associato ad un oggetto SqlConnection, preventivamente connesso alla sorgente dati, e può contenere una query di selezione (per leggere i dati dal database) o una query d'azione (per aggiornare i dati), impostata tramite la proprietà CommandText.
Dopo aver impostato l'oggetto SqlCommand, ed aver impostato la query mediante la proprietà CommandText, si deve utilizzare uno dei relativi metodi Execute:

- **ExecuteNonQuery** si utilizza per inviare una query d'azione, e restituisce il numero di record coinvolti. Permette attività di manipolazione di dati, corrispondenti alle istruzioni SQL di Insert, Update e Delete.
- **ExecuteReader** si utilizza per inviare una query di selezione, e restituisce l'oggetto DataReader che consente di accedere al set dei risultati (resultset).
- **ExecuteScalar** si utilizza per inviare una query di selezione, e restituisce un singolo

MIGRARE DA ACCESS A SQL SERVER 2005

La migrazione è veramente semplice. Esiste un tool apposito nel menu strumenti di Access 2000 chiamato Upsizing Tool che vi guiderà passo passo all'upgrade. Se invece disponete di un Access 97 sarà necessario scaricare l'apposito tool, l'indirizzo di riferimento è il seguente: http://www.microsoft.com/products/developer/officedeveloper/access/prodinfo/aut97dat.htm. Nelle nostre prove non abbiamo verificato nessun problema di perdita di dati nell'upsizing. Il tool si compone di pochi passi che vi invitano a selezionare le tabelle origine e destinazione. Al termine dell'esportazione otterrete il nuovo database e un file di LOG che contiene l'esito dell'esportazione tabella per tabella. In un solo caso abbiamo incontrato una difficoltà, ovvero con una tabella di un db Access chiamata "file" in tal caso viene generato un conflitto di tipi e l'esportazione non va a buon fine. In questi rari casi è sufficiente rinominare le tabelle che generano il conflitto per risolvere il problema.

> DOVE POSIZIONARE IL FILE

4 Nella parte inferiore, della sezione General, spostando la barra di scorrimento verso destra alla voce path possiamo cambiare il percorso del file fisico costituente il database, con valori diversi da quelli proposti per default (per il nostro esempio lasciamo quelli proposti). Possiamo, inoltre, cambiare la dimensione iniziale (ad es. 5 MB), ed attivare l'opzione di crescita automatica delle dimensioni del database. In generale è consigliabile impostare un aumento in termini percentuali

> LE PROPRIETÀ

5 Nella sezione Opzioni possiamo modificare alcune proprietà relative ad esempio all'update del database, per il nostro esempio lasciamo quelle di default.

valore risultato di un'istruzione Select (da usare, ad esempio, se il risultato è il frutto di query con clausole di aggregazione come count o sum).

L'oggetto SqlCommand espone diverse proprietà, tra queste:

- **CommandText** permette di impostare l'istruzione SQL della query o la stored procedure da eseguire sull'origine dati.

- **CommandType** permette di impostare il valore che indica in che modo interpretare il tipo di query impostato nella proprietà CommandText, può assumere i valori: Text, StoredProcedure

- **Connection** rappresenta l'oggetto SqlConnection associato all'istanza corrente dell'oggetto SqlCommand.

- **Parameters** contiene la collezione di tutti i parametri principali associati all'oggetto SqlCommand.

INSERIRE O MODIFICARE UN RECORD

Per eseguire una query di azione sul database, tipicamente si deve:

- Creare un oggetto SqlCommand con le proprietà fondamentali CommandText e SqlConnection.

- Eseguire il comando Sql tramite il metodo ExecuteNonQuery, che restituisce il numero di record coinvolti dall'istruzione

Inseriamo, ad esempio, una nuova persona nel database MiaRubrica.
La prima operazione è quella di costruire la stringa corrispondente all'istruzione Sql di inserimento (Insert)

```
Dim QuerySql As String
QuerySql = "INSERT INTO Persona (CodicePersona,
                Nome,Cognome,NumeroTelefono)VALUES
 ( 1,'Federica','Buono','098222222')"
```

Successivamente si deve creare un oggetto SqlCommand ponendo la proprietà CommandText pari alla stringa che definisce la query, e la proprietà SqlConnection pari all'oggetto objConnection creato nell'esempio precedente:

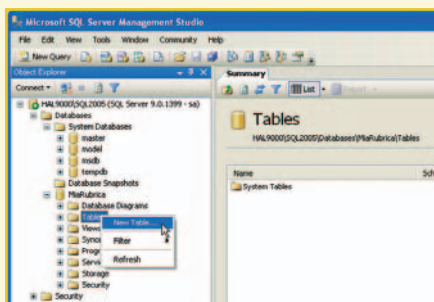
```
Dim ObjCommand As New SqlCommand(QuerySql,
                                ObjConnection)
```

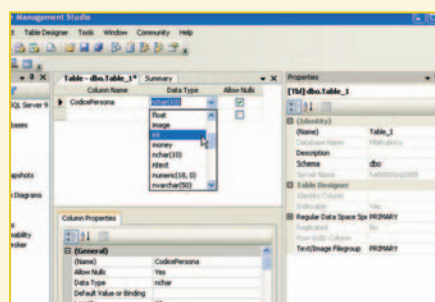
Infine si deve eseguire il comando Sql tramite il metodo ExecuteNonQuery, che restituisce il numero di record coinvolti dall'istruzione:

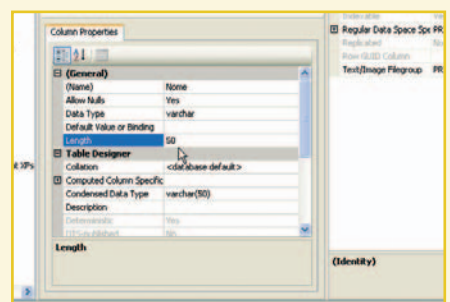
```
Dim NumeroRecord As Integer
NumeroRecord = ObjCommand.ExecuteNonQuery()
```

CREAZIONE DI UNA TABELLA

Una volta creato il nostro Database dobbiamo definire le entità fondamentali che permetteranno di utilizzarlo: le tabelle

> SCEGLIERE IL DB

1 Selezioniamo il Database di esempio (MiaRubrica), espandiamo il ramo e clicchiamo con il tasto destro del mouse sulla voce Tables. Dal menù contestuale possiamo selezionare il menu: New Table.

> CREARE LE COLONNE

2 Cliccando nel campo Data Type il sistema mostra un menù con tutti i tipi di dati disponibili in SQL Server, da cui possiamo selezionare il tipo desiderato (ad esempio int oppure float).

> DIMENSIONARE I DATI

3 In base al tipo selezionato al passo precedente, apparirà, nella parte inferiore della finestra, il campo Lenght che permetterà di specificare la lunghezza, o la lunghezza massima del campo

Seguendo lo stesso schema possiamo modificare un record della tabella persona, costruendo la stringa corrispondente all'istruzione Sql di aggiornamento (update)

```
Dim QuerySql As String
QuerySql = "update Persona set Nome ='Sara',
        Cognome='Buono', NumeroTelefono='556688'
where CodicePersona=1"
Dim ObjCommand As New SqlCommand(QuerySql,
                                    ObjConnection)
Dim NumeroRecord As Integer
NumeroRecord = ObjCommand.ExecuteNonQuery()
```

Per eliminare un record, si deve seguire sempre lo stesso schema utilizzando l'istruzione SQL di cancellazione *(delete Persona where CodicePersona=1)*.

ESEGUIRE QUERY DI INTERROGAZIONE

Per interrogare il database e leggere i dati si deve:

- Creare un oggetto SqlCommand come abbiamo visto in precedenza.
- Eseguire la query Sql tramite il metodo ExecuteReader, ed assegnarlo ad un oggetto SqlDataReader per leggere il set dei risultati una riga per volta.

Ad esempio, per eseguire una ricerca di tutte le persone presenti in MiaRubrica, si può scrivere:

```
Dim QuerySql As String
QuerySql = "select * from persona"
Dim ObjCommand As New SqlCommand(QuerySql,
                                    ObjConnection)
Dim ObjReader As SqlDataReader =
                         ObjCommand.ExecuteReader()
```

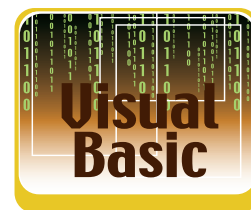
Analizziamo in dettaglio come utilizzare l'oggetto SqlDataReader per leggere il set di risultati ottenuto.

L'OGGETTO SqlDataReader

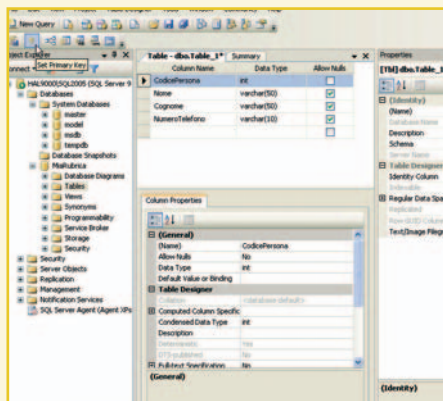
L'oggetto SqlDataReader viene utilizzato per leggere un set dei risultati di tipo forward-only da un database SQL Server, a seguito di una query d'interrogazione. A differenza degli oggetti visti in precedenza non è possibile utilizzare un costruttore per creare un oggetto SqlDataReader, ma è invece necessario chiamare il metodo ExecuteReader dell'oggetto SqlCommand. Tra le proprietà ed i metodi esposti dall'oggetto SqlDataReader segnaliamo:

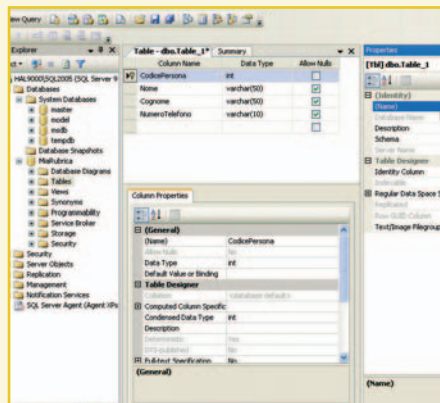
- FieldCount contiene il numero di colonne del record corrente.
- IsClosed contiene un valore, pari a True, se

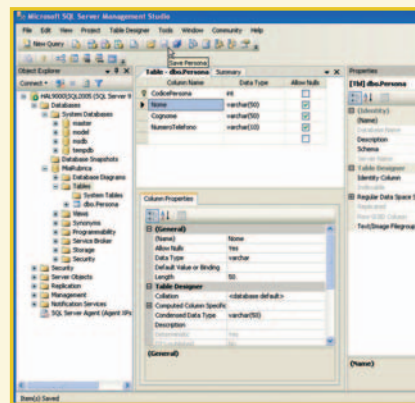
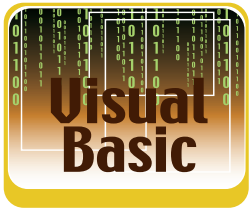
Visual Basic

NOTE

DOVE SI INSTALLA SQL SERVER?

SQL Server Management Studio viene installato per impostazione predefinita in C:\Program Files\Microsoft SQL Server\90\Tools\Binn\VS Shell\Common7\IDE.

> LE CHIAVI PRIMARIE

6 Dopo aver completato la definizione dei campi, possiamo impostare la chiave primaria della tabella selezionando il campo (CodicePersona) e cliccando sull'icona a forma di chiave sulla barra degli strumenti, così come le altre impostazioni

> SCEGLIERE IL NOME

7 A questo punto possiamo assegnare un nome significativo alla tabella, ad esempio Persona, operando sulla finestra di dialogo di destra, la finestra Properties, e modificando il valore del campo Name nel modo più opportuno

> SALVARE TUTTO

8 Infine dobbiamo salvare la tabella appena creata, cliccando sull'icona a forma di dischetto sulla barra degli strumenti . A questo punto il nostro DB è creato e nel caso del formato MDF lo troveremo nella directory opportuna

il DataReader è chiuso.

- RecordsAffected contiene il numero di record modificati, inseriti o eliminati dall'esecuzione dell'istruzione SQL.

- Close chiude l'oggetto SqlDataReader e rilascia le risorse allocate.
- GetName contiene il nome della colonna con l'indice specificato.

- IsDbNull contiene un valore , pari a True, se la colonna contiene valori nulli.

- Read sposta l'oggetto SqlDataReader al record successivo. Restituisce un valore pari a True se sono presenti altri record da elaborare, oppure False se si è arrivati alla fine del set dei risultati

- GetValue contiene il valore della colonna con l'indice specificato nel suo formato nativo

Al posto del metodo GetValue è possibile utilizzare uno dei numerosi metodi Get fortemente tipizzati, come GetString o GetDecimal. In pratica c'è un metodo Get per ogni tipo di dati gestito da VB oppure un metodo GetSql per ogni tipo di dati gestito da Sql Server. Questi metodi particolari dovrebbero essere sempre adottati, in quanto evitano gli errori di conversione provocati dalla perdita di precisione e generano codice più veloce.

MIGLIORIE AL CODICE

Per mantenere il codice il più compatto possibile, si può utilizzare l'istruzione Imports che semplifica l'accesso alle classi, eliminando la necessità di digitare in modo esplicito il nome completo del namespace che le contiene. Le istruzioni Imports devono sempre essere scritte nella parte superiore del file nel quale si vogliono utilizzare, prima di qualunque altro codice. Per queste ragioni, in tutti gli esempi di codice, è ragionevole assumere l'inserimento delle seguenti istruzioni Imports:

```
Imports System.Data
Imports System.Data.SqlClient
```

UTILIZZARE L'OGGETTO SqlDataReader

Per iterare sui singoli record di un set di risultati, utilizzando l'oggetto SqlDataReader, è sufficiente:

- invocare il metodo Read per avanzare al record successivo nel set di risultati
- verificare il relativo valore di ritorno per controllare se esistono altri risultati (nel caso sia pari a True) oppure si è arrivati alla fine e non c'è nessun'altra riga da leggere (se è False).

Grazie al funzionamento del metodo Read è possibile creare un ciclo While..End While basato sull'oggetto SqlDataReader. Per completare l'esempio precedente, in cui si cercano tutti le persone presenti in MiaRubrica, possiamo scrivere il seguente codice che mostra un messaggio per ogni persona in archivio:

```
Dim CodicePersona, Nome, Cognome,
                              NumeroTelefono As String
While ObjReader.Read()
   CodicePersona = ObjReader.GetInt32(0)
   Nome = ObjReader.GetString(1)
   Cognome = ObjReader.GetString(2)
   NumeroTelefono = ObjReader.GetString(3)
   MessageBox.Show(CodicePersona & " " & Nome &
           " " & Cognome & " " & NumeroTelefono)
End While
```

Mentre si utilizza l'oggetto SqlDataReader, non è possibile eseguire alcuna operazione sull'oggetto SqlConnection associato, per questo è importante chiudere l'oggetto SqlDataReader quando non esistono più righe da elaborare, in modo da rilasciare le risorse (lato client e lato server) e rendere la connessione nuovamente disponibile per altri comandi:

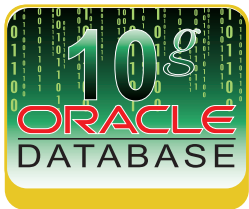
```
ObjReader.Close()
```

SQL SERVER 2005 E VISUAL STUDIO 2005

La nuova versione di Sql Server si caratterizza per un'integrazione estrema con il Visual Studio 2005 Express. All'interno dell'ambiente di fatto esistono una serie di Wizard che si pongono come punto nevralgico nella creazione di una qualunque applicazione.
Di fatto Sql Server è talmente integrato che l'ambiente stesso di programmazione diventa anche il query builder e il project manager del database. Inoltre il provider è sufficientemente strutturato da consentire con semplici operazioni di Drag & Drop di realizzare procedure che altrimenti richiederebbero molte righe di codice. Ad esempio la creazione di una maschera di inserimento dati si riduce quasi esclusivamente a trascinare i campi del database sulla form.

Luigi Buono

ORACLE 10g XE: NUOVO E GRATIS

ORACLE INTRODUCE UNA NUOVA VERSIONE DEL SUO DATABASE DOTATA DI FUNZIONALITÀ PER VELOCIZZARE E OTTIMIZZARE LO SVILUPPO. E RENDE IL TUTTO LIBERAMENTE UTILIZZABILE. UNA RIVOLUZIONE NEL CAMPO DEI DATABASE…

Oracle è da sempre sinonimo di database di qualità, così come MySQL e Postgres sono sinonimi di database gratuiti. Per colmare questa "lacuna", Oracle ha deciso di rendere la versione 10g XE, express edition, gratuita da scaricare e utilizzare nelle applicazioni, senza bisogno di pagare licenze. Si può quindi integrare la potenza di uno dei database più noti e autorevoli al mondo nei progetti che realizziamo. Inoltre, la versione 10g ha delle caratteristiche innovative rispetto alla precedente 9i che permettono di sfruttare meglio le operazioni che si compiono normalmente sui database. Uno degli elementi che possiamo usare subito, è l'introduzione delle espressioni regolari nel linguaggio SQL e PL/SQL. Usando una sintassi simile a quella di molti linguaggi di programmazione e a Unix, permettono di eseguire query più facili ed efficaci lavorando direttamente sui dati. Migliorano le operazioni su oggetti di grandi dimensioni come i LOB. Considerando l'internazionalizzazione dei dati, ci sono nuove funzioni per eseguire l'ordinamento a seconda della lingua scelta. E nuovi tipi di dato, binary_float e binary_double, per avere migliori precisione e performance. Operazioni di flashback per ritornare a stati precedenti di righe e tabelle o per vedere quando una riga è stata modificata o ripristinare una tabella prima di una drop. L'elenco delle funzionalità introdotte è tale che nella New Features Guide ci sono 67 pagine solo per loro!

ESPRESSIONI REGOLARI

Una volta passate in rassegna alcune delle nuove caratteristiche di Oracle 10g, si può passare all'azione vedendo quanto siano facili e potenti da usare. Alcune funzionalità sono migliorie di operazioni già esistenti, altre sono novità assolute: è questo il caso delle espressioni regolari. Spesso capita di dover cercare tra i dati, tutti i valori che contengono alcune lettere. Ad esempio, una query come:

```
SELECT campo FROM tabella WHERE campo LIKE '%fra%';
```

potrà restituire valori come francia, afragolese, affranto ecc. Cioè tutti i valori che contengono "fra". Le espressioni regolari servono proprio a potenziare questo meccanismo per vedere se certi termini sono presenti, in che posizione compaiono oppure sostituirli. Il grande vantaggio è che si può lavorare direttamente nel database senza perdere tempo a prelevare il dato, modificarlo o verificarlo con i linguaggi di programmazione come java o c# e poi, magari, inserirlo nuovamente nel database.

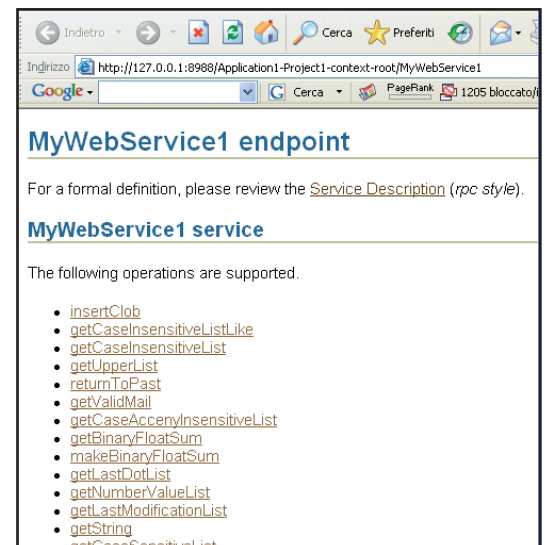



Fig. 1: Web services dell'applicazione

Per prendere confidenza con le espressioni regolari si può vedere la prima funzione realizzata nel progetto che consente di prelevare tutti i dati di un campo che iniziano con la lettera maiuscola. L'applicazione, scritta in java, prevede una classica esecuzione di una query usando Statement e ResultSet.

```
//OracleDAO. getUpperList()
rs = stmt.executeQuery("SELECT * " + "FROM prova
    " + "WHERE REGEXP_LIKE(prova, '^[[:upper:]]')");
```

Come si può notare, accanto agli elementi caratteristici delle query, ci sono REGEXP_LIKE che richiama

le espressioni regolari e poi la combinazione di alcuni elementi. ^[[:upper:]] significa che il primo valore, specificato dal simbolo ^, deve essere di tipo upper, cioè maiuscolo. In modo del tutto analogo si possono cercare stringhe che contengano caratteri alfanumerici, solo numeri, spazi o punteggiatura. Tutto ciò è una grande innovazione rispetto al vecchio SQL che ha potenzialità decisamente più limitate. Se invece si deve considerare l'ultimo valore della stringa allora si usa $. Mettendo insieme queste considerazioni, e cercando di ottimizzare il tempo di esecuzione, facendo lavorare il database, si passa al metodo successivo che seleziona tutti i dati che sono delle mail sintatticamente valide. In una forma semplice, una mail è una sequenza del genere: renato@rossi.it con un dominio separato dal punto e con una @. Anche in questo caso l'uso delle espressioni regolari consente di scrivere una query semplice:

```
//OracleDAO.getValidMailList()
rs = stmt.executeQuery("SELECT * " + "FROM prova
 " + "WHERE REGEXP_LIKE(prova, " + "'[[:alnum:]]+
                        @[[:alnum:]]+ [.][[:alnum:]]')");
```

nella quale si identificano gli elementi fissi come la @ e quelli parametrici come [[:alnum]] che considerano valori alfanumerici. Altre espressioni regolari permettono di trovare il punto in cui l'espressione stessa è soddisfatta, come ad esempio l'indice dell'ultimo punto di una stringa. Nell'esempio di prima, *renato@rossi.it*, il punto si trova in posizione 13.

```
//OracleDAO.getLastDotList()
rs = stmt.executeQuery("SELECT REGEXP_INSTR" +
                    "(prova, '([.][[:alnum:]]+)$') " +
                              "as ind from prova");
```

In questo caso, oltre ai simboli già visti, c'è anche il + che significa che l'elemento precedente, un carattere alfanumerico, deve comparire almeno una volta.

LARGE OBJECTS

Un'altra situazione comune è quella di dover memorizzare, nei database, grandi quantità di dati sotto forma di molti kb di testo, immagini, video o suoni. Oracle supporta questo tipo di memorizzazione con l'introduzione dei LOB, *Large Objects*, che possono essere di 3 tipi: BLOB, *Binary LOB*, CLOB, *Character LOB*, e NCLOB, *National Character LOB*. I nomi suggeriscono il formato che viene usato per memorizzare il dato, binario o testuale, e il tipo di nazionalità del carattere specificato dal database. Un caso che si può presentare è quello di registrare un file xml nel database e il problema avviene alla soglia dei 32765 byte. Infatti, a seconda se il file aveva dimensione maggiore o minore di questo valore si dovevano usare 2 metodi di 2 oggetti diversi (java.sql .PreparedStatement e oracle.sql.CLOB). Ora queste chiamate sono diventati trasparenti grazie all'uso di una proprietà della connessione.

```
//OracleDAO.prepareConnection()
Properties props = new Properties();
props.put("SetBigStringTryClob", "true");
```

Il resto del codice è, a questo punto, del tutto identico all'inserimento di una normale stringa di dimensione qualsiasi.

```
//OracleDAO.insertClob(String str)
pstmt = con.prepareStatement(
"INSERT INTO prova (prova, clobcolumn)
                      VALUES('test clob1', ?)");
pstmt.setString(1, str);
pstmt.executeUpdate();
```

Con i metodi prepareStatement e setString si imposta la query per inserire nel campo clobcolumn della tabella prova il valore str che è una String di dimensioni qualsiasi. A volte capita di dover registrare un file, ad esempio xml, di dimensioni variabili: in questo modo non serve più discriminare via codice sulla dimensione del file: fa tutto Oracle.

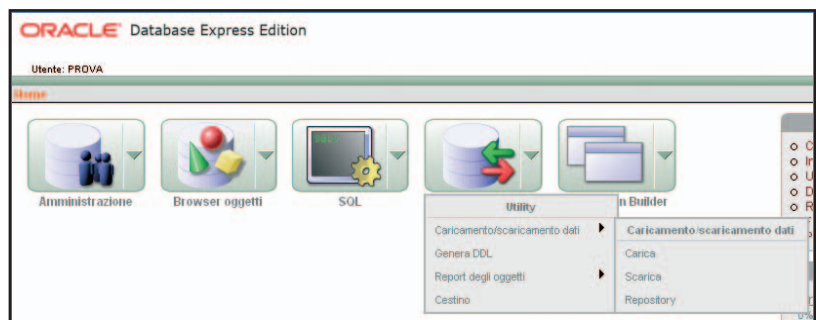



Fig. 2: *La console di amministrazione di Oracle xe*

GLOBALIZZAZIONE E RICERCHE

Con questo termine, o con internazionalizzazione, si intende il processo di rendere il programma adatto per qualsiasi lingua, considerando anche i caratteri speciali propri di ogni nazione, come ad esempio la "ä". Quando si devono effettuare ricerche sui dati immettendo parte del nome oppure si devono ritornare liste ordinate alfabeticamente, allora possono essere utili le funzioni di globalizzazione. Consideriamo il caso in cui dobbiamo prendere una lista di valori ordinata alfabeticamente. Se non si imposta nessuna condizione specifica, la lista restituita presenterà prima i valori numerici, poi le stringhe che iniziano con la lettera maiuscola, poi quelle con la lettera minuscola, infine lettere speciali accentati e

URL PER INIZIARE

Il sito della Oracle contiene le informazioni necessarie per lavorare al meglio con il database 10g.
Ci sono le informazioni generali e i link dove poter scaricare il database.
http://www.oracle.com/database/index.html

di lingue estere. In questo modo l'ordinamento è case sensitive.

10
Ale
CIAO
CIao
Ciao
ale
ciao
ciao stai come
àle
äle

Usando l'SQL standard si possono ottenere risultati migliori, ma che non coprono tutti i casi possibili. Con le funzionalità di globalizzazione si possono valutare molteplici soluzioni che vanno incontro a tutte le esigenze. Iniziando dal caso più facile, si possono ordinare i valori in modalità case insensitive.

10
ale
Ale
Ciao
CIao
CIAO
ciao
ciao stai come
àle
äle

In questo modo quello che si vede è una lista nella quale ci sono sottoliste di stringhe che iniziano tutte con la stessa lettera, indipendentemente dal fatto che sia maiuscola o minuscola. Resta fuori ancora qualcosa. Si possono vedere gi ultimi valori: iniziano con lettere accentate, non importa se sono lettere dell'alfabeto italiano o meno. A volte si devono poter mettere anche queste parole nel gruppo di quelle che iniziano con "a". Ecco allora che una piccola variazione della funzione restituisce l'effetto desiderato per ottenere un ordinamento sia case insensitive che accent insensitive.

10
äle
Ale
àle
ale
CIAO
CIao
Ciao
ciao
ciao stai come

Considerando gli obiettivi, si può andare a vedere il codice che genera questi tre risultati. I tre metodi si trovano tutti nella classe OracleDAO e hanno nomi che ricordano la ricerca che viene effettuata, cioè una lista "case sensitive", "case insensitive" e "case e accent insensitive".

```
//OracleDAO.getCaseSensitiveList()
rs = stmt.executeQuery("select prova from prova ");
...
//OracleDAO.getCaseInsensitiveList()
rs = stmt.executeQuery("select prova from prova " +
        "order by nlssort(prova,'nls_sort=binary_ci')");
...
//OracleDAO.getCaseAccentInsensitiveList()
rs = stmt.executeQuery("select prova from prova " +
        "order by nlssort(prova,'nls_sort=binary_ai')");
```

Come si può vedere dal codice, la funzione che permette l'ordinamento è nlssort che prende il campo sul quale effettuare l'ordinamento e il tipo di ordinamento che è specificato da nls_sort. Alcuni tipi di ordinamento sono ci, case insensitive e ai, accent insensitive. Volendo si possono impostare parametri specifici per ogni nazionalità. Un modo alternativo per eseguire le query impostando prima il tipo di ordinamento consiste nel modificare le impostazioni del database.

```
alter session set NLS_SORT=BINARY_CI;
```

Questa istruzione può essere lanciata dal database oppure inserita in una query apposite e lanciata dal programma java. Infine si può considerare il caso che unisce alcune caratteristiche impiegate fino ad ora. Una situazione frequente prevede la ricerca di una stringa in modalità case insensitive. Ad esempio la ricerca di tutte le parole che contengono "ci" sia nel caso di lettere maiuscole che minuscole. Usando solo le espressioni regolari otterremmo tutte le righe con "ci" minuscole. Fondendo insieme il potere espressivo delle espressioni regolari con le funzionalità di globalizzazione, si ottiene il risultato desiderato.

```
//OracleDAO.getCaseInsensitiveListLike(String name)
rs = stmt.executeQuery("select prova from prova " +
      "where REGEXP_LIKE(prova,'" + name + "','i')");
```

Come si può vedere dal codice, è tutto molto semplice: una combinazione di REGEXP_LIKE con la globalizzazione data dal parametro "i" alla fine che indica la ricerca case insensitive.

FLASHBACK

Lavorando con i database si conosce bene il rischio di modifiche ai dati. A volta questi vengono cambia-

SUL WEB

URL PER LA DOCUMENTAZIONE

Il sito della Oracle permette di accedere a tutta la documentazione immaginabile sulle varie versioni del database. Si trovano guide specifiche per amministratori, sviluppatori, aspetti specifici come i *LOB, OLAP, SQL, XML.*
http://www.oracle.com/pls/db10g/portal.portal_demo3?selected=3

ti in modo non corretto dall'applicazione o dall'amministratore o magari cancellati e poi può diventare impossibile ricostruire la situazione o la storia di un record o di una transizione. Per poter gestire eventualità del genere, sin da Oracle 9i ci sono delle operazioni che consentono di recuperare il vecchio stato dei record. Con la versione 10g le funzionalità sono aumentate e diventa facile riportare il database, una tabella o una singola riga al momento desiderato. Una delle informazioni che si possono avere è data e ora dell'ultima modifica delle righe della tabella.

```
//OracleDAO.getLastModificationList()
rs = stmt.executeQuery("select
        scn_to_timestamp(ora_rowscn), prova " +
                              "from prova ");
while (rs.next()) {
  lastModificationList.add(rs.getString(1)+ ", "
                              + rs.getString(2));
}
```

In questo metodo ci sono due elementi caratteristici di Oracle: *scn_to_timestamp* e *ora_rowscn*. *ora_rowscn* è una pseudocolonna, come *rowid*, che definisce il *System Change Number*, SCN, cioè il numero del cambiamento più recente. Il valore viene incrementato ogniqualvolta avviene un'operazione di commit. In questo modo si può gestire meglio anche la transazionalità di un operazione: infatti si può prendere questo valore prima di eseguire le operazioni, eseguirle senza fare commit, verificare che nessun altro abbia cambiato la tabella vedendo se il valore è sempre lo stesso, infine eseguire il commit. Prima di *ora_rowscn*, per verificare se vi erano stati cambiamenti si doveva memorizzare i dati e confrontarli tutti. L'altro elemento usato è *scn_to_timestamp* che, come suggerisce il nome, serve per convertire il valore scn in un timestamp. Il metodo, nel caso specifico, serve per prendere tutte le righe della tabella e la loro data di ultima modifica. I valori sono poi concatenati e aggiunti in una lista per essere visualizzati. Con questo metodo possiamo sapere quindi tutte le modifiche alle righe. Se invece si vuole recuperare lo stato della tabella, riprendendo i dati così come erano alcuni minuti prima, si può usare il metodo *returnToPast(String minuteAgo)*.

```
// OracleDAO.returnToPast(String minuteAgo)
pstmt = con.prepareStatement(
  "INSERT INTO provaflashback (prova)
      (SELECT prova FROM prova AS OF TIMESTAMP
                (SYSTIMESTAMP - INTERVAL '"
                + minuteAgo + "' MINUTE))");
pstmt.executeUpdate();
```

Con un *PreparedStatement* si può inserire nella tabella *provaflashback* tutti i valori della tabella prova come erano alcuni minuti prima. Il parametro dei minuti è impostabile dall'utente ed è *minuteAgo*. In questo caso si usano sempre le potenzialità del flashback unite alla gestione delle date con systimestamp. Per poter eseguire operazioni di flashback si possono quindi usare sia i timestamp sia il valore *scn:* la differenza non è casuale e Oracle consiglia di usare scn perché ha una precisione maggiore. Il meccanismo che permette di effettuare questi salti a stati precedenti si basa sulla scrittura delle caratteristiche delle tabelle in file di flashback log e, quando si eseguono le operazioni i dati vengono rimessi per differenza da quelli presenti, ottimizzando quindi le risorse. Se ad esempio è cambiata solo una riga, sarà solo quella ad essere modificata e tutto il resto resterà immutato. Le operazioni di recupero possono essere compiute anche sull'intero database.

```
FLASHBACK [DEVICE TYPE = <device type>]
                                DATABASE
TO [BEFORE] SCN = <scn>
  TO [BEFORE] SEQUENCE = <sequence> [THREAD
                           = <thread id>]
      TO [BEFORE] TIME = '<date_string>'
```

Dalla sintassi si può apprezzare come il flashback del database può essere eseguito usando sia il valore *scn* sia una data.

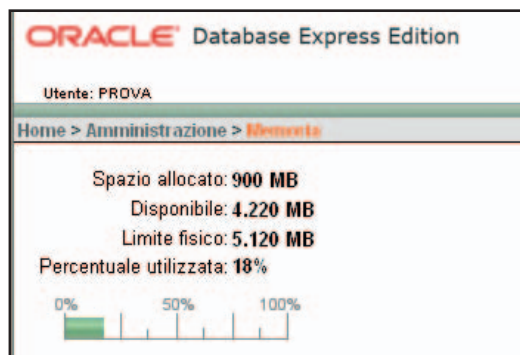



Fig. 3: **La console mostra la memoria utilizzata**

DATI NUMERICI

I tipi di dato sono tanti e le caratteristiche che i valori numerici possono assumere nelle applicazioni tendono a essere infiniti. Ecco perché si tenta di regolare i tipi numerici, come suggerisce IEEE che definisce lo standard per i valori binari in virgola mobile e che sono seguiti anche dai linguaggi di programmazione come java. In questo contesto fanno il loro ingresso 2 nuovi tipi di dato: *BINARY_FLOAT* e *BINARY_DOUBLE*. Usando questi tipi, anziché il generico *NUMBER*, si ottimizzano le pre-

stazioni in termini di velocità di calcolo, spazio occupato e grandezza del valore memorizzato. E come per *NUMBER* c'è la funzione di conversione *TO_NUMBER*, così ora si possono utilizzare *TO_BINARY_FLOAT* e *TO_BINARY_DOUBLE*. Naturalmente questi valori sono compatibili con le normali operazioni numeriche e di aggregazione come sum, avg, sin, cos. Collegati ai nuovi tipi di dato ci sono anche delle costanti che possono essere usate nelle query o nei constraint, quando si definiscono le tabelle. Anche questi sono del tutto simili a quelli usati nei linguaggi di programmazione: *IS (NOT) INFINITE* e *IS (NOT) NAN*, che verificano se un numero è o meno infinito o non è un numero. Mettendo tutto insieme si possono creare situazioni nelle quali i valori restituiti sono anomali. Infatti, se eseguendo la somma di valori *BINARY_FLOAT* il risultato è ancora un valore di dimensioni accettabili, allora lo possiamo memorizzare in tabella. Se invece il valore è troppo grande si potrebbe avere un valore di ritorno non numerico.

BINFLOAT
Inf

Questo è il risultato che si ottiene usando la console di amministrazione di Oracle; mentre Infinity è quello che si ottiene eseguendo il metodo dell'applicazione.

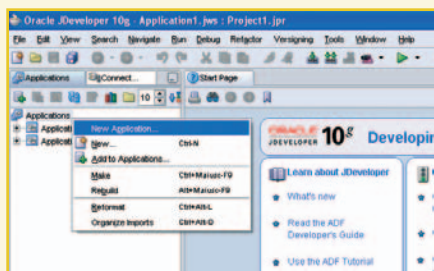
```
//ORacleDAO.getBinaryFloatSum()
rs = stmt.executeQuery("select binfloat from prova " +
        "where prova = 'somma'");
```

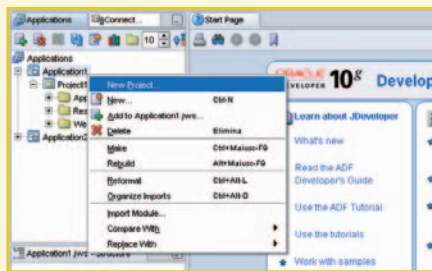
Un modo per evitare queste situazioni è quello di utilizzare IS NOT INFINITE e IS NOT NAN nella definizione della tabella. La sintassi di questi vincoli utilizza la parola chiave check e si può trovare accanto ai vincoli primary key, not null o unique.

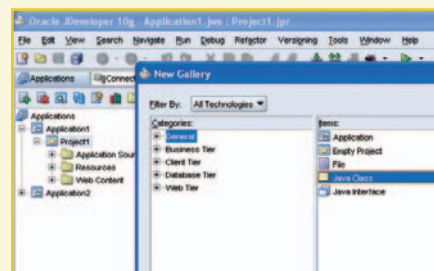
```
create table floating_point_table (
    floatNotNull binary_float constraint flt_null not null,
    floatUnique binary_float constraint flt_unq unique,
        floatCheck binary_float constraint
        flt_chk check (floatCheck is not NaN ) ,
        doubleCheck binary_double constraint
        dbl_chk check (doubleCheck is not infinite) ,
        floatPrimaryKey binary_float constraint
        flt_ floatPrimaryKey primary key);
```

FACCIAMOLO CON JDEVELOPER

Primi passi per creare un Web Services utilizzando lo strumento gratuito di Oracle

> CREARE UNA NUOVA APPLICAZIONE

1 Per prima cosa si crea una nuova applicazione.

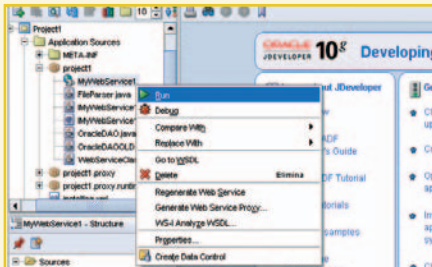
> CREARE UN NUOVO PROGETTO

2 Poi un nuovo progetto tramite clic con il tasto sinistro del mouse.

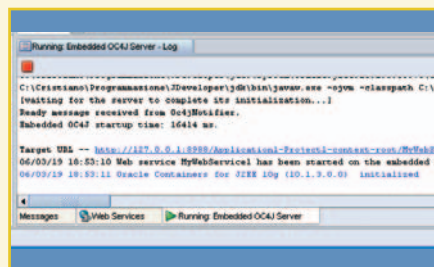
> CREARE UNA NUOVA CLASSE

3 Creazione di una classe che servirà per esporre i web services.

> IMPLEMENTARE IL CODICE

```
public interface IMyWebService1 {
    public String getString();
    public String[] getArrayString();
    public String[] getUpperList();
    public String[] getValidMail();
    public String[] getLastDotList();
    public String[] getNumberValueList();
    public String insertClob();
    public String[] getCaseSensitiveList();
```

4 All'interno della classe appena creata potremo aggiungere il codice

> FAR PARTIRE L'APPLICATION SERVER

5 Possiamo testare il codice con uno strumento interno al prodotto

> APERTURA DEL BROWSER

6 Cliccare sull'url che compare per aprire il browser ed eseguire l'applicazione.

Questa definizione della tabella mostra come il campo floatCheck debba essere not NaN, cioè debba essere numerico e doubleCheck non debba essere infinito. Abbiamo visto con l'esempio di prima cosa vuol dire e quando accade che un valore è infinite, mentre il valore NaN si ottiene quando si eseguono operazioni matematiche che restituiscono risultati indeterminati come ad esempio 0/0 oppure infinito/infinito.

L'APPLICAZIONE

Fin qui abbiamo parlato delle novità di Oracle 10g riferendoci all'applicazione realizzata senza dire molto sull'organizzazione del software e sugli strumenti che Oracle mette a disposizione dei programmatori per velocizzare lo sviluppo del codice. L'applicazione realizza dei web services che esportano tutte le funzionalità viste in precedenza ed è divisa in 3 file: 1) WebServiceClass, che esporta i servizi e li realizza; 2) OracleDAO, che accede al database per gestire le informazioni; 3) FileParser, che esegue il parsing di un file, anche xml, per eseguire l'inserimento del CLOB. WebServiceClass si occupa di coordinare tutte le chiamate a OracleDAO e FileParser. La stringa di connessione che contiene i parametri sul tipo di driver usato, thin, sul nome o indirizzo della macchina, localhost, sulla porta e sul database, 1521 e xe è utile se si vuole cambiare server o le altre caratteristiche.

```
//OracleDAO.prepareConnection()
String url = "jdbc:oracle:thin:@localhost:1521:xe";
```

Può essere anche utile e riusabile l'oggetto FileParser che, in modo veramente semplice, prende in ingresso un file xml, che nel nostro caso va messo nella directory principale, e lo trasforma in una stringa da mettere poi nel database.

```
//FileParser
try {
   File file =
      new File("\\installlog.xml");
   FileReader in = new FileReader(file);
   int c;
   sb = new StringBuffer();
   while ((c = in.read()) != -1){
      sb.append((char)c);
   }
} catch (IOException ex) {
      System.out.println("Exception reading file" + ex);
}
```

Quello che avviene è la creazione di un oggetto di tipo File che legge un xml e lo converte in StringBuffer, salvo errori di tipo input/output.

L'AMBIENTE ORACLE

Di Oracle possiamo usare il database 10g xe, gratuito, scaricabile da internet, e installarlo sul computer che vogliamo. Il processo di installazione è molto facile e, una volta completato, permette di accedere ai comandi, sia grafici sia a riga di comando, per lanciare e fermare il database e per far partire la console di amministrazione. Da questa si può creare il nuovo utente di prova e le tabelle per eseguire l'applicazione. In allegato ci sono i file con le definizioni delle tabelle usate e alcuni dati per eseguire le operazioni dell'applicazione. Da console è facile eseguire caricamento/scaricamento dati e export /import delle tabelle. Oracle permette lo sviluppo di applicazioni java con l' ide JDeveloper versione 10g che è gratuito.

L'ambiente di sviluppo consente di creare facilmente i web services e di lanciare velocemente l'applicazione, senza bisogno di deploy, grazie all'application server integrato in JDeveloper che si chiama OC4J Server. Quindi, per provare tutto il progetto, non serve installare altro.

CONCLUSIONI

Le caratteristiche di Oracle 10g sono molte e rivestono ambiti diversi. Si va da quelle per gli amministratori a quelle per gli sviluppatori a quelle che consentono di migliorare le prestazioni e di ottimizzare l'utilizzo delle risorse.

Gli obiettivi che si è posta Oracle con la versione 10g, grid, sono di aumentare la sicurezza, le performance e la scalabilità usando un insieme di server per la gestione dei database. Potenzialmente, si possono aggiungere o rimuovere computer o memoria ed il database automaticamente alloca le risorse sfruttando il massimo delle risorse disponibili. Ci sono poi miglioramenti per la fase di tuning e di monitoring.

Le variazioni a SQL che consentono agli sviluppatori di ottimizzare le operazioni da eseguire sui dati. Basti pensare che si stima che l'80% della logica applicativa è processare stringhe. Ecco allora l'introduzione delle espressioni regolari e dei LOB. E anche le migliorie per il recupero dei dati modificati o la globalizzazione. E anche l'introduzione dei nuovi tipi di dato del tutto simili a quelli già presenti nei linguaggi di programmazione. Infine forse la caratteristica che potrebbe far propendere per un database Oracle: la sua gratuità per la versione xe. Se poi si aggiunge anche un ambiente che facilita lo sviluppo come JDeveloper, ecco allora che ci sono tutte le basi per poter usare Oracle per confezionare un'applicazione dall'inizio alla fine.

Cristiano Bellucci

URL PER JDEVELOPER

Il sito della Oracle fornisce la documentazione anche per il suo ide, JDeveloper.

http://www.oracle.com/technology/obe/obe1013jdev/index.htm

E anche l'url per scaricarlo gratuitamente.

http://www.oracle.com/technology/software/products/jdev/index.html

PHP SPEDISCE LE NEWSLETTER

IMPLEMENTEREMO UN'INTERFACCIA WEB CHE CONSENTA AD UTENTI AUTORIZZATI DI INTERAGIRE CON IL SISTEMA, E UN ENGINE CHE SPEDISCE I CONTENUTI IN MODO UN PO' PARTICOLARE....

Prima di ogni cosa chiariamoci le idee su quello che vogliamo realizzare. Da un punto di vista delle pure funzionalità il nostro sistema dovrà:

- Autenticare un amministratore
- Consentire agli utenti autenticati di aggiungere una newsletter
- Permettere di aggiungere/rimuovere gli utenti
- Inviare le email in modo schedulato di modo che non si affolli il mail server

In realtà la gestione degli utenti dovrà funzionare in modalità mista. In questo articolo non affronteremo l'argomento, ma gli utenti dovrebbero potersi iscrivere/rimuovere dalla newsletter tramite un'interfaccia web pubblica. Daremo per scontato che questa parte del sistema sia già implementata e funzionante.

GLI STRUMENTI

Come sempre facciamo in questa serie di articoli su PHP, ci faremo aiutare dalle classi Pear per agevolare l'interfacciamento con i database e per gestire l'autenticazione, ed utilizzeremo Smarty per la gestione dei template e l'interfaccia Web.
Al solito per installare le classi Pear che ci servono digiteremo da una console di sistema

```
pear install auth
pear install db
```

Nel primo caso ci verrà richiesto qualche modulo di storage supplementare, l'installazione è facoltativa, in questo articolo utilizzeremo un'autenticazione base che non richiede particolari attenzioni.
Per quanto riguarda smarty invece, avremo cura di scaricarlo dal sito smarty.php.net, scompattarlo e copiare in una directory raggiungibile dal percorso di include del php, a tal proposito verificate nel php.ini quali sono le vostre directory di inclusione e adattatele in modo che siano conformi alle vostre esigenze. Infine per fare funzionare bene smarty, avremo bisogno di creare le seguenti directory all'interno della root del nostro progetto

```
templates
templates_c
configs
```

la directory template_c dovrà essere accessibile in scrittura, conterrà infatti i file compilati di smarty. Anche in questo caso accenniamo brevemente a questa caratteristica ma non è nostro scopo approfondire il discorso dei template in questo momento. Poiché siamo in fase di definizione del progetto creiamo anche una directory "kernel" al cui interno inseriremo le classi che ci servono per la gestione del progetto.

L'AUTENTICAZIONE

Prima di ogni cosa creiamo un file config.php all'interno della nostra directory kernel. Sarà qui che inseriremo le variabili globali, come ad esempio la stringa di connessione al database. Il nostro file config.php conterrà

```
<?php
include_once("DB.php");
require_once("Auth/Auth.php");
require_once('Smarty.class.php');
$server = "localhost";
$database = "mailbot";
$username = "mailbot";
$password = "mailbot_pwd";
$dbh = DB::connect("mysql://$username:
                    $password@$ server/$database");
$params = array( "dsn" => "mysql://$username:
                    $password@$server/$database" ,
"table" => "auth" ,
"usernamecol" => "username" ,
"passwordcol" => "password",
'db_fields'=>"*"
);
```

```
$a = new Auth("DB", $params, "loginFunction");
?>
```

in mysql creiamo un database chiamato mailbot

```
mysqladmin -uroot -ppassword create mailbot
```

e aggiungiamo i permessi per l'utente mailbot con password mailbot_pwd

```
mysql -uroot -ppassword
grant all privileges on mailbot.* to mailbot identified
                                          by 'mailbot_pwd'
```

e creiamo la tabella auth

```
mysql -umailbot -pmailbot_pwd mailbot
CREATE TABLE `auth` (
  `ID_USER` bigint(20) NOT NULL auto_increment,
  `username` varchar(50) NOT NULL default '',
  `password` varchar(32) NOT NULL default '',
  `ROLE` bigint(20) NOT NULL default '1',
  PRIMARY KEY  (`ID_USER`,`username`),
  KEY `password` (`password`)
)
```

A questo punto siamo pronti per generare il nostro file index.php che in un primo momento avrà questa forma:

```
<?
        require_once('kernel/config.php');
 $a->start();
        if ($_GET['action'] == "logout") {
                $a->logout();
        $a->start();
        }
        if ($a->getAuth()) {
        $smarty = new Smarty;
        function list_newsletter() {
        global $smarty,$db;
                $smarty->display('index.tpl'); }
        switch ($_POST['operazione']) { {
          case "list_newsletter":
          list_newsletter();
          break;
          default:list_newsletter();
          break;   }
   }
?>
```

e aggiungiamo alla tabella auth un utente fittizio “admin” con password “admin” tanto per poter fare delle prove. Allo stesso modo creiamo un file fittizio index.tpl all'interno della directory templates al cui interno inseriremo una stringa casuale, anche in questo caso per poter testare il funzionamento del modulo di autenticazione. Puntiamo il browser su http://localhost/mailbot/index.php e se tutto è andato a buon fine visualizzeremo la form per il login e saremo in grado di visualizzare il contenuto del file index.tpl solo se riusciremo ad autenticarci. Se tutto funziona correttamente il nostro modulo di autenticazione può considerarsi terminato.

LA HOME PAGE

Nella nostra home page desideriamo che dopo aver effettuato l'autenticazione, compaia una lista delle newsletter già inviate o pronte per l'invio, un link per la gestione degli utenti, e un link per l'aggiunta di una nuova directory. Creiamo la tabella in mysql che dovrà contenere le newsletter:

```
CREATE TABLE `newsletter` (
`ID_NEWSLETTER` BIGINT NOT NULL
                         AUTO_INCREMENT PRIMARY KEY ,
        `TITOLO_NEWS` VARCHAR( 255 ) NOT NULL ,
                `CORPO_NEWS` LONGTEXT NOT NULL
) TYPE = MYISAM
```

Poi creiamo un nuovo file all'interno della directory kernel, che si chiamerà newsletter.php. In questo file inseriremo le classi che ci serviranno per gestire il progetto. Inizialmente il suo contenuto sarà:

```
<?
  include_once('kernel/config.php');
  class newsletter {
     function store($titolo,$news) {
        global$dbh;
        $query = "INSERT INTO newsletter (
                TITOLO_NEWS,CORPO_NEWS) values
                                ('$titolo','$news')";
        if ( DB::isError( $demoResult = $dbh->query(
                                     $query ) ) ) {
           echo DB::errorMessage($demoResult);
        }}}
?>
```

Il file index.php dovrà essere modificato aggiungendo un include per file newsletter.php

```
require_once('kernel/newsletter.php');
```

aggiungiamo poi due funzioni per la gestione dell'inserimento

```
function store_newsletter() {
                        global $_POST;
  $titolo= $_POST["titolo_news"];
  $contenuto= $_POST["content_news"];
  $newsletter= new newsletter();
  $newsletter->store($titolo,$contenuto);
  list_newsletter();
```

```
}
function add_newsletter() {
   global $smarty;
    $smarty->display('add_newsletter.tpl');
}
```

e un case per gestire gli eventi

```
switch ($_POST['operazione']) {
   case "add_newsletter": add_newsletter();
   break;
   case "store_newsletter": store_newsletter();
   break;
   case "list_newsletter": list_newsletter();
   break;
   default:list_newsletter();
   break;
}
```

non ci resta che sistemare i template. In index.tpl ci dovrà essere un combobox che ci mostra l'azione da compiere:

```
<form method="post" action="index.php"">
     <select name="operazione">
          <option value="add_newsletter">Aggiungi
                                          news</option>
     </select>
<input type="submit">
</form>
```

utilizzando questa form verrà richiamato ancora una volta index.php con il campo “operazione” posto a “add_newsletter”. Questo fa si che quando index.php viene richiamato il controllo passi alla funzione add_newsletter() che mostra il contenuto di add_newsletter.tpl che a sua volta conterrà:

```
<html>
<body>
   <form method="post" action="index.php"">
          <input type="hidden" name="operazione"
                            value="store_newsletter">
   <table cellspacing="2" class="fz">
<tr><td><label for="titolo_news">Titolo news:
                                        </label></td>
<td><input type="text" name="titolo_news"
                  maxlength="56"  tabindex="1" /></td>
</tr>
<tr><td><label for="content_news">Corpo
                                    news:</label></td>
<td><textarea name=content_news
                                rows="15"  cols="50">
scrivi qui
</textarea>
<input type="submit">
</td>
```

```
</tr> </table> </form> </body></html>
```

a questo punto operazione cambia in “store_newsletter” e quando il controllo tornerà al file index .php verrà richiamata la funzione store_newsletter() che provvederà tramite la classe che abbiamo creato in precedenza a salvare la newsletter sul nostro database.
Il meccanismo di modifica e rimozione di una newsletter segue lo stesso meccanismo e non ne parleremo in questo articolo. Aggiungeremo invece alla classe un metodo fetchAllNewsLetter che ci consentirà di visualizzare l'elenco delle newsletter inserite. Per cui all'interno di newsletter.php aggiungeremo:

```
  function fetchAllNewsLetter() {
       global $dbh;
       $stmt="select * from newsletter)";
       $dettagli_newsletter= $dbh->getAll($stmt,
                                 DB_FETCHMODE_ASSOC);
       return $dettagli_newsletter;
  }
```

la funzione list_newsletter di index.php diventerà

```
function list_newsletter() {
   global $smarty,$db;
   $newsletter=new newsletter();
   $allnews = $newsletter->fetchAllNewsLetter();
   $smarty ->assign( "allnews" , $allnews);
   $smarty->display('index.tpl');
}
```

e potremo modificare index.tpl per listare tutte le newsletter presenti in archivio

```
<form method="post" action="index.php"">
<select name="operazione">
   <option value="add_newsletter">Aggiungi
                                   news</option>
      </select>
<input type="submit">
</form>
<br>
{section name="i" loop=$allnews}
{$allnews[i].TITOLO_NEWS}
<form method="POST" action="index.php">
<input type=hidden name="operazione"
                            value="invia_news">
<input type=hidden name="id_news"
              value="{$allnews[i].ID_NEWSLETTER}">
<input type="submit" value=submit>
</form>
{/section}
</form>
```

Notate anche che abbiamo aggiunto un pulsante “submit” accanto al nome di ogni newsletter.

MIGLIORIE

Lo scheletro qui proposto potrebbe essere migliorato aggiungendo una funzione che garantisce di inviare un numero limitato di funzioni nel tempo, di modo che il mailserver non sia eccessivamente affollato. Per realizzare questa funzione si potrebbe aggiungere al db utenti un campo “news da inviare” e un campo “prossima_news” e poi agire su uno scheduler che a tempi determinati invia le email aggiornando i due campi

Questo pulsante una volta cliccato avrà la funzione di richiamare ancora una volta index.php e passargli in post il valore “operazione=invia_news”, sempre in post gli passerà il valore id_news della news da postare. E' qui che potete eventualmente agire per aggiungere altri pulsanti che serviranno a loro volta per modificare/eliminare una news. E con questo anche il modulo di amministrazione delle news è terminato. Non ci resta che passare al modulo send che servirà per inviare fisicamente la newsletter

IL MODULO SEND

Come sempre prima di ogni cosa creiamo una tabella mysql che conterrà il nome, il cognome, l'indirizzo email della persona che dovrà ricevere la newsletter

```
CREATE TABLE `user` (
`ID_USER` BIGINT NOT NULL AUTO_INCREMENT
                                    PRIMARY KEY ,
`NOME` VARCHAR( 100 ) NOT NULL ,
`COGNOME` VARCHAR( 100 ) NOT NULL ,
`EMAIL` VARCHAR( 100 ) NOT NULL
) TYPE = MYISAM ;
```

Diamo per scontato che in qualche modo questa tabella sia stata riempita con valori opportuni. Tipicamente sarà riempita tramite la registrazione di un utente sul sito. Ovviamente trascuriamo in questa sede tutte le note sulla privacy etc. Siamo certi che siate nelle condizioni legali di poter inviare a questi utenti la vostra newsletter.
A questo punto creeremo un nuovo file send.php all'interno del kernel che conterrà le classi per l'invio dell'email. Il nostro send.php inizialmente conterrà il seguente codice:

```
require_once("kernel/newsletter.php");
include_once('kernel/config.php');
 class servermail {
   function send($ID_NEWS) {
      $newsletter=new newsletter();
      $newsletter_to_send=$newsletter->
                        fetchNewsLetterById($ID_NEWS);
      $usertosend=$this->fetchuser();
       foreach ($usertosend as $user) {
         $from_name = "NewsLetter User";
         $from_email = "ffarnesi@edmaster.it";
         $subject = $newsletter_to_send[0][
                                        'TITOLO_NEWS'];
         $body = "------=
                    MIME_BOUNDRY_message_parts\n";
         $body .= "Content-Type: text/plain; charset=
                                      \"iso-8859-1\"\n";
         $body .= "Content-Transfer-Encoding:
                                   quoted-printable\n";
         $body .= "\n";
         $body = $newsletter_to_send[
                         0]['CORPO_NEWS']."\n\n";
         $headers = "From: $from_name
                                  <$from_email>\n";
         il($user['EMAIL'],$subject,$body,$headers);
       }}
    function fetchuser() {
      global $dbh;
       $stmt="select EMAIL from user";
       $userlist= $dbh->getAll($stmt,
                              DB_FETCHMODE_ASSOC);
       return $userlist;
     } }
?>
```

chiaramente in index.php avremo

```
switch ($_POST['operazione']) {
            case "invia_news":
            invia_news();
            break;
            ....
```

e ancora

```
            function invia_news() {
               global $_POST;
               $ID_NEWS=$_POST["id_news"];
               $sender= new servermail();
               $sender->send($ID_NEWS);}
```

ovviamente dovremo creare in newsletter.php il metodo fetchNewsLetterByID($ID_NEWS); che sarà il seguente

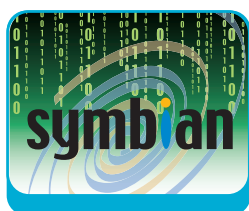
```
function fetchNewsLetterById($ID_NEWS) {
   global $dbh;
   $stmt="select * from newsletter where ID
                    _NEWSLETTER='$ID_NEWS'";
   $dettagli_newsletter= $dbh->getAll($stmt, DB_
                              FETCHMODE_ASSOC);
   return $dettagli_newsletter; }
```

ed il gioco è fatto. Il metodo send non fa altro che costruire una lista degli utenti tramite un metodo interno fetchuser, recuperare il testo della newsletter, costruirla e inviarla.

CONCLUSIONI

In questo articolo abbiamo suggerito uno schema estremamente personalizzabile. Per adattarlo alle vostre esigenze tutto quello che è necessario fare è modificare i vari template ed aggiungere gli eventi nello switch di index.php, ovviamente non dimenticate di costruire i metodi opportuni nel kernel.

BLUETOOTH TRAMITE L'OBEX PROTOCOL

METODI, CLASSI, TEORIA E TECNICHE NECESSARI AD IMPLEMENTARE IL TRASFERIMENTO DEI DATI IN DISPOSITIVI MOBILI. ANALIZZEREMO UN ESEMPIO DI NOKIA PER IMPARARE COME LE NOSTRE APPLICAZIONI POSSANO INTERAGIRE FRA LORO

In questo articolo affronteremo la problematica riguardante l'invio e la ricezione di dati, attraverso il protocollo Bluetooth con il sistema operativo per telefonia mobile, Symbian. Spiegheremo come effettuare una semplice trasmissione dati utilizzando il protocollo *OBEX*. La scelta dell'OBEX deriva dal fatto che è facilmente adattabile anche al trasferimento tra un device mobile ed un PC. L'esempio utilizzato in questo articolo, è basato su quello fornito insieme al Symbian SDK, dal colosso Svedese Nokia, e precisamente il *Btobjectexchange*. Questo perché si tratta di un esempio molto didattico che esaurisce a fondo tutte le problematiche relative allo scambio dati con bluetooth. Il primo passo consisterà dunque nello scaricare l'SDK adatto al vostro cellulare. Potete farlo all'indirizzo *http://www.forum.nokia.com/*. La nostra applicazione dovrà sostanzialmente seguire questa sequenza logica:

- Ricerca di periferiche bluetooth nel raggio d'azione del telefonino
- Connessione alla periferica con cui vogliamo scambiare i dati
- Instanziazione dei vari componenti che permettono la comunicazione
- Invio del classico messaggio "Hello World" da un telefono all'altro.

In tutto questo, al di là dei vari sistemi che servono per ricercare, connettere, far comunicare i vari sistemi, il protocollo di trasporto che si occuperà fisicamente di trasferire i dati da un telefono all'altro sarà l'Obex.

IL PROTOCOLLO OBEX

Questo particolare standard, fornisce le funzioni indispensabili per trasferire interi oggetti o *chunks* di dati (interi blocchi di dati binary),da un device all'altro. Si tratta di un protocollo utilizzato non solo per la comunicazione BlueTooth ma anche con altri mezzi di trasporto, ad esempio iRDA, ovvero il tra-

Fig.1: *Sequence Diagram di un'applicazione OBEX*

sferimento dati con gli infrarossi. Il sistema operativo Symbian supporta pienamente Obex e le API che lo gestiscono sono inglobate in 5 classi definite nel file di inclusione *"obex.h"*:

- *TObexBluetoothProtocolInfo*
- *TObexConnectInfo*
- *MObexServerNotify*
- *CObexClient*
- *CObexServer*

La classe *TObexBluetoothProtocolInfo* estende la classe *TobexProtocolInfo.* Come gia detto OBEX può essere utilizzato con un gran numero di layer fisici di trasporto, iRDA piuttosto che Bluetooth per esempio. Per ciascuno di questi layer fisici abbiamo bisogno di una classe specifica derivata da *TobexProtocolInfo,* nel nostro caso ovviamente, utilizzeremo *TobexBluetoothProtocolInfo.* La sua interfaccia pubblica completa è la seguente:

```
class TObexBluetoothProtocolInfo : public
TObexProtocolInfo
{  public:
    TRfcommSockAddr iAddr;
    TBuf<60> iTransport; // dalla classe
```

Conoscenze richieste
C++

Software
L'SDK della Nokia per il Symbian serie 60. Una connessione Bluetooth su PC ed un cellulare provvisto di connessione Bluetooth o di due telefoni con sistema operativo Symbian.

Impegno

Tempo di realizzazione

```
                                        TObexProtocolInfo
};
```

La classe *TobexBluetoothProtocolInfo* eredita da *TobexProtocolInfo* un oggetto di tipo *iTransport* che conterrà un nome che indica il protocollo di trasporto da utilizzare: *RFCOMM* per OBEX, *irTyniTP* per iRDA, e la estende con un oggetto di tipo *TrfcommSockAddr* che conterrà l'indicazione sulla porta o sul canale di comunicazione da usare in una connessione con il device remoto. La classe *TObexConnectInfo* contiene informazioni relative alla versione dell'Obex, e altri dati che possono essere utili per avere informazioni generiche sui pacchetti. Ne riportiamo la costruzione per completezza

```
class TObexConnectInfo
{   public:
      inline TObexConnectInfo();
      inline TUint8 VersionMajor() const;
      inline TUint8 VersionMinor() const;
      ...
};
```

La classe *MObexServerNotify* implementa tutte le funzioni necessarie a gestire gli eventi che possono occorrere durante la comunicazione. Ad esempio avverte l'applicazione client in caso di fallimento della connessione, oppure quando viene richiesta una nuova connessione, o ancora gestisce l'evento relativo all'arrivo del primo pacchetto valido e così via. Il codice che la implementa è il seguente

```
class MObexServerNotify
{  friend class CObexServer;
   private:
      virtual void ErrorIndication(TInt aError) =0;
      virtual void TransportUpIndication() =0;
      virtual void TransportDownIndication() =0;
      virtual TInt ObexConnectIndication(const
                              TObexConnectInfo& aRemoteInfo,
                                    const TDesC8& aInfo) =0;
      virtual void ObexDisconnectIndication(const
                                         TDesC8& aInfo) =0;
      virtual CObexBufObject* PutRequestIndication() =0;
      virtual TInt PutPacketIndication() =0;
      virtual TInt PutCompleteIndication() =0;
      virtual CObexBufObject* GetRequestIndication(
                     CObexBaseObject *aRequiredObject) =0;
      virtual TInt GetPacketIndication() =0;
      virtual TInt GetCompleteIndication() =0;
      virtual TInt SetPathIndication(const
                          CObex::TSetPathInfo& aPathInfo,
                                    const TDesC8& aInfo) =0;
      virtual void AbortIndication() =0; };
```

Una breve descrizione degli eventi gestiti è la seguente:

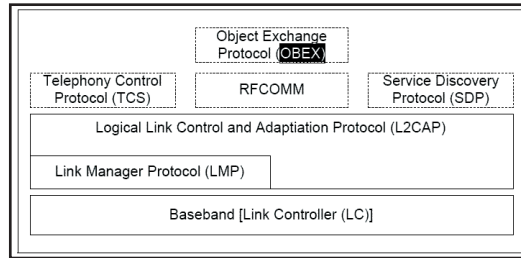
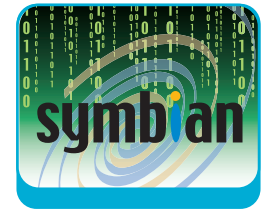
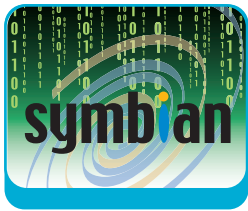



Fig.2: *Il protocollo OBEX*

- **ErrorIndication** – viene eseguito quando si verifica un errore che costringe la connessione Obex ad essere terminata. Non vengono presi in considerazione errori che non terminano la connessione. Il suo valore di ritorno è un intero che è definito nella lista codici errore del Symbian.
- **TransportUpIndication** – viene chiamata dal sistema operativo, quando un'applicazione Client richiede la connessione ad un Server. Questo però non indica che una sessione OBEX sia realmente iniziata, ma semplicemente che una connessione è stata richiesta.
- **TransportDownIndication** – viene chiamata dal sistema operativo, quando una delle due parti termina la connessione
- **ObexConnectIndication** – Il sistema operativo chiama questa funzione quando viene stabilita una connessione OBEX. All'interno di *ObexConnectIndication* viene mantenuto il parametro aRemoteInfo contiene le informazioni relative alla versione di OBEX utilizzata dalla macchina client che ha richiesto la connessione.
- **ObexDisconnectIndication** – Questa funzione è chiamata quando una connessione OBEX termina correttamente. Non viene richiamata quando l'applicazione client si disconnette inaspettatamente.
- **PutRequestIndication** – Viene chiamata quando viene ricevuto il primo pacchetto valido. Tuttavia questa funziona non effettua alcuna operazione di parsing del pacchetto. Semplicemente indica che un pacchetto valido è stato ricevuto, a questo punto si può decidere cosa il sistema operativo ne deve fare. Il valore restituito da questa funzione sarà un oggetto *CobexBufObject* derivato da *CobexBaseObject* che fornirà tutte le indicazioni sul come eventualmente scrivere il dato ricevuto in un file.
- **PutPacketIndications** – Questa funzione segue sempre una *PutRequestIndication* e contiene informazioni sulla ricezione del pacchetto. Può essere utilizzata ad esempio per fornire un'indicazione sullo stato delle operazioni.
- **PutCompleteIndication** – Quando questa funzione è chiamata, l'applicazione sarà sicura di aver ricevuto tutti i pacchetti OBEX inviati dal device remoto. In caso la connessione si interrompesse o vada in un qualsiasi errore, questa fun-

DOVE TROVARE L'ESEMPIO

L'esempio a cui ci riferiamo, può essere trovato, una volta installato l'SDK, al seguente path:

<Directory di installazione>\7.0s\Series60\series60Ex\BTObjectExchange\

zione non verrà chiamata.

- **GetRequestIndication** – Questa funzione viene eseguita sull'applicazione Server quando l'applicazione Client ne richiee un particolare oggetto.
- **GetPacketIndication** – Una volta iniziata la trasmissione mediante il protocollo OBEX, questa funzione verrà chiamata ad ogni pacchetto inviato.
- **GetCompleteIndication** – Questa funzione viene chiamata quando la trasmissione dati è completata, in modo da permettere all'applicazione Client di effettuare la deallocazione di tutte le azioni.
- **AbortIndication** – Questa funzione viene chiamata quando il device Client ha interrotto la trasmissione in modo controllato.

Finita questa lunga introduzione teorica siamo pronti per passare alla pratica. Vediamo come utilizzare le informazioni fin qui acquisite per realizzare i nostri scopi

IDE CONSIGLIATI

Il C++ Builder X versione Mobile 1.5, contiene componenti non visuali per il collegamento e il trasferimento tramite protocollo Bluetooth.

LA RICERCA DEI DISPOSITIVI

Il Symbian offre, per la ricerca dei devices disponibili, due metodologie principali:

1. Il Client istanzia una richiesta di identificazione; in questo modo può rendersi conto di quanti dispositivi fisici sono disponibili nel raggio di azione delle tecnologie scelte (Bluetooth, iRDA)
2. Il Client utilizza l'interfaccia del device Bluetooth *(Bluetooth Device Selection UI)*. In questo caso è un vero e proprio Plug-In che si occupa di interrogare tutti i dispositivi presenti. Dopo aver rilevato le risposte, le mostra all'utente in un Dialog Box e successivamente sarà poi selezionato il device desiderato.

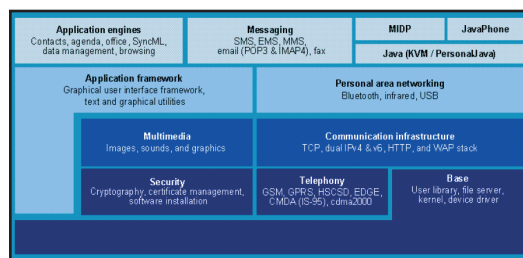



Fig.3: ***Architettura del Symbian***

In questo articolo, ci occuperemo del secondo metodo sia perché è quello utlizzato dall'esempio citato nell'introduzione, sia perché lo riteniamo quello più semplice e produttivo. La *"Device Selection UI"*, è un Plug-in per la classe *RNotifier* la quale ne permette l'utilizzo di vari tipi predisposti dal sistema

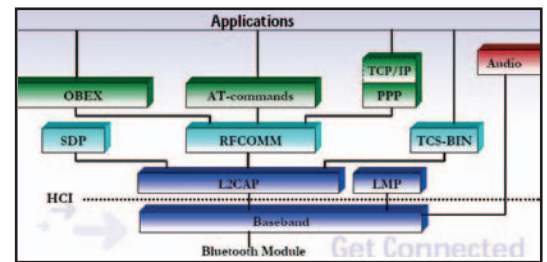



Fig. 4: ***Schema del Bluetooth***

operativo. Ora, supponiamo che stiate utilizzando l'applicazione *"Rubrica"* e che da questa applicazione vogliate attivare la ricerca dei dispositivi bluetooth presenti nel vostro raggio d'azione. Sarebbe l'applicazione *"Rubrica"* a dover implementare la dialog box per la visualizzazione dei dispositivi perché questa possa comparire al di sopra dell'interfaccia principale. Questo è ovviamente scomodo, di fatto ogni applicazione dovrebbe contenere una definizione della dialog box se si volesse fare in modo che essa apparisse sempre in primo piano rispetto all'applicazione che l'ha chiamata. Il problema viene risolto facendo in modo che l'applicazione "Client" instanzi una richiesta ad un Thread Server che gira in background che si occupa sia di recuperare la lista dei dispositivi presenti sia di costruire la dialog box e proiettarlo in primo piano rispetto all'applicazione Client che ha instanziato la richiesta. Questo thread viene denominato *"Notifier Server"*. La classe che consente la connessione ad un *"Notifier Server"* è la *Rnotifier*. Un esempio di come viene utilizzata la classe *Rnotifier* per connettersi al Thread, e instanziare la richiesta di creazione della dialog box con tutti i dispositivi trovati è presente nel metodo *CBTDiscoverer::SelectDeviceL* contenuto nel sorgente *btdiscover.cpp* presente nell'SDK della Nokia. Il primo passo è connettersi al Thread che gira in background, per farlo possiamo usare

```
RNotifier not;
User::LeaveIfError(not.Connect());
```

Una volta connessi, viene inizializzato il plugin per la selezione dei device tramite un oggeto di tipo *TBTDeviceSelectionParamsPckg*.

```
TBTDeviceSelectionParamsPckg selectionFilter;
```

Questo oggetto contiene un'istanza della classe *TBTDeviceSelectionParams*. Inizialmente questa classe è riempita e quindi inizializzata con valori di default.

```
TRequestStatus status;
```

RNotifier è progettata per usare un *Active Object* per mandare al processo Client, una notifica quando l'utente seleziona un Item. La *RNotifier* è chiamata

per mostrare il Dialog di selezione dei dispositivi Bluetooth. Tale DialogBox, che si andrà successivamente a creare, mostrerà all'utente la lista dei dispositivi Bluetooth attivi all'interno della rete locale *PicoNet.* A questo punto, la classe *RNotifier* esegue il codice per raggiungere tutti i dispositivi presenti.
Nella chiamata alla funzione principale, il primo parametro sarà il *TRequestStatus.* Il secondo è una costante intera rappresentante il *Dialog* della selezione dei dispositivi bluetooth. Il terzo parametro è il *Selection Filters* e successivamente l'ultimo parametro è il parametro di risposta contenente il dispositivo Bluetooth scelto dall'utente nel *DialogBox.*

```
not.StartNotifierAndGetResponse( status,
            KDeviceSelectionNotifierUid, selectionFilter,
                                        aResponse );
```

Poiché il processo di selezione è asincrono, cioè si attende che l'utente scelga il dispositivo dal Dialog Box, l'applicazione Client dovrà utilizzare la funzione *WaitForRequest* per mettere in pausa il Thread dell'applicazione stessa fino a quando l'*RNofitier* non avrà terminato il suo lavoro. In questo modo l'utente dovrà necessariamente fare la scelta per il dispositivo nel Dialog di selezione, dato che l'applicazione si fermerà fino a quando la scelta non avverrà. Questo come detto, mediante il metodo *WaitForRequest*

```
User::WaitForRequest(status);
```

La variabile *Status,* è utilizzata anche per determinare l'esito di ritorno dall'esecuzione dell'*RNotifier,* in particolare ecco come controllare gli stati ed i codici di errore:

```
if (status.Int() == KErrNone)
{  if (aResponse().IsValidDeviceName())
   { success = ETrue; }
   else
   { iReporter.Error(_L("Failed to connect")); } }
else
{  iReporter.Error(_L("No device selected"));
}
```

L'applicazione Client ha finito con l'oggetto *RNotifier* e quindi, procediamo alla sua deallocazione. Per far questo, utilizziamo i metodi della *RNotifier* per deallocare sia il DialogBox, sia chiudere la connessione con il Server. Il tutto avverrà se il Server della *RNotifier* non ha nessun'altra connessione in atto, liberandone così tutte le risorse.

```
not.CancelNotifier(KDeviceSelectionNotifierUid);
not.Close();
```

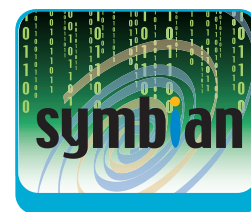
La funzione a questo punto, ritorna il valore

```
return success;
}
```

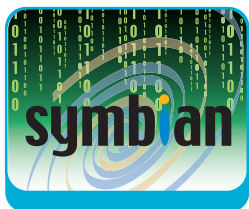
LA CLASSI COBEXCLIENT E COBEXSERVER

Queste classi fanno sì che l'applicazione Client comunichi con l'OBEX Server, trasmettendogli o ricevendo dati. L'esempio è riportato nell'interfaccia pubblica della *COBexClient,* della quale ne vediamo una parte del codice:

```
class CObexClient : public CObex
{  public:
    IMPORT_C ~CObexClient(); IMPORT_C static
                CObexClient* NewI(TObexProtocolInfo&
                                aObexProtocolInfoPtr);
    IMPORT_C void Connect(TRequestStatus& aStatus);
    IMPORT_C void Put(CObexBaseObject& aObject,
                                TRequestStatus& aStatus);
    IMPORT_C void Get(CObexBaseObject& aObject,
                                TRequestStatus& aStatus);
    IMPORT_C void Abort();
...
};
```

Affinché l'applicazione Client possa connettersi con un device server, dovrà, una volta creato un *CObexClient,* conoscerne l'indirizzo specificandolo insieme al protocollo da usare per la connessione. Tutte le informazioni, sono contenute nell'oggetto *TObexBluetoothProtocolInfo,* visto in precedenza. Il Client, una volta ottenuta l'istanza della classe OBEX, potrà connettersi con il device OBEX remoto, e potrà richiedere o mandare un oggetto OBEX. Tutto ciò

FRAMEWORK AGGIUNTIVI

In aiuto allo sviluppatore di applicazioni Bluetooth, vi è un progetto che semplifica la realizzazione di applicazioni che sfruttano tale protocollo. Una delle più affermate è senza alcun dubbio il *Framework Cobain*. Il *Cobain* fornisce tutte le API di cui necessita un'applicazione Bluetooth per Symbian. Ecco un sorgente di esempio per l'invio di un file tramite le *Cobain API*:

```
// create the API facade
CCobainLayer *cobain =
                    CCobainLayer::NewL();
// fetch Bluetooth driver (Cobain will
                support other carriers too)
MNetworkDriver *driver = cobain->
                GetDriverL(EBluetooth);
// fetch peers synchronously - for demo
// purposes, normally we'd use
// asynchronous, of course
TPeerList *peerlist = driver->
                GetPeersL(0x123456);

// connect to the first peer found
CNetworkPeer *peer = (*peerlist)[0];
RCobainSocket *socket = peer->
                        ConnectL();
// send some data
_LIT8(KData, "hello, world!");
socket->SendL(&KData);
// close & cleanup
socket->Close();
delete driver;
delete cobain;
```

sarà possibile utilizzando le tre funzioni asincrone, *Connect(), Get(), Put()*. L'operazione, o meglio la trasmissione, potrà essere interrotta mediante la funzione *Abort()*. Il *CObexServer* fa in modo che un'applicazione Client, estenda la disponibilità dei servizi OBEX, ad altri device.

```
class CObexServer : public CObex
{
public:
    IMPORT_C static CObexServer* NewI(
            TObexProtocolInfo& aObexProtocolInfoPtr);
    IMPORT_C ~CObexServer();
    IMPORT_C TInt Start(MObexServerNotify* aOwner);
    IMPORT_C void Stop();
    inline TBool IsStarted();
    inline TOperation CurrentOperation() const;
...
};
```

Come per la classe *CObexClient*, quando allochiamo la classe *CObexServer,* l'applicazione Client deve fornire un oggetto *TObexProtocolInfo,* che specificherà al server, tutte le informazioni dette di "ascolto". Queste informazioni includono il protocollo da utilizzare, la porta, oppure quando si sviluppano applicazioni Bluetooth come in questo caso, il canale su cui il server rimane in ascolto. Una volta che l'oggetto del Server è stato creato, il Server potrà partire. Per far ciò, si utilizza la funzione *Start()* che, una volta chiamata, obbligherà il Client a dare un'istanza della classe *MObexServerNotify* che fa parte anch'essa delle classi dette *Observer.* Il sistema operativo ritornerà tutti gli eventi rilevanti attraverso la classe *CObexServer.* Il Server potrà essere fermato, utilizzando la funzione *Stop();*.
Le funzioni *ISStarted* e *CurrentOperation*, controlleranno invece lo stato del Server.

L'ESEMPIO BTOBJECTEXCHANGE DI NOKIA

Come detto nella nostra introduzione, per poter capire bene l'utilizzo dello standard Bluetooth mediante l'*OBEX Protocol*, ci riferiremo all'esempio fornito dalla Nokia con l'SDK della Serie 60. L'esempio è il *BTObjectExcange Example*.
Le impostazioni che servono ad un *CObexServer* per una trasmissione tramite connessione Bluetooth, sono pressoché identiche per qualsiasi altra connessione che si avvale di un altro protocollo. Tutti i parametri delle impostazioni, derivano dal protocollo *RFCOMM* mediante l'utilizzo delle *Bluetooth Sockets* API, di cui anche l'OBEX si avvale.

IMPOSTIAMO IL SERVER

Le impostazioni dell'OBEX, vengono definite nella funzione AvailableServerChannelL, questa funzione è presente nel file sorgente *ObjectExchangeServer.cpp,* dell'esempio *BTObjectExchangeExample*. Nella stessa classe, troviamo la funzione *InitialiseServerL()* che è parte dell'OBEX Server e che permette di impostare la porta di "ascolto" e quindi di far partire il server.

```
Void CobectExchangeServer::InitialiseServerL()
{ ...
  // Cerca la porta di attesa dati
  TInt channel = AvailableServerChannelL();
```

Una volta ottenuto il canale, l'applicazione provvederà al settaggio di tutte le impostazioni di sicurezza del protocollo Bluetooth. Questo avviene mediante l'utilizzo della funzione *SetSecurityChannelI();*.
Successivamente, per far partire effettivamente il server, viene creato l'oggetto *TObexBluetoothProtocolInfo* ed inizializzato impostandone il canale con le informazioni ottenute in precedenza.
Inoltre, viene impostato il nome del protocollo richiesto, mediante una stringa che verrà utilizzata per specificarne il nome. In questo caso, la stringa è *KServerTransportName:*

```
  TObexBluetoothProtocolInfo obexProtocolInfo;
obexProtocolInfo.iTransport.Copy(KServerTransportName);
obexProtocolInfo.iAddr.SetPort(channel);
```

Creato l'oggetto *TObexBluetoothProtocolInfo,* il *COBexServer* può essere istanziato e successivamente potrà esere avviato:

```
  iObexServer = CobexServer::NewL(
                              obexProtocolInfo);
```

L'avvio del server, viene effettuato mediante la fuzione Start();:

```
  iObexServer->Start(this);
  iAdvertiser->StartAdvertisingL(channel);
  iAdvertiser->UpdateAvailabilityI(ETrue);
...
}
```

Adesso dobbiamo impostare il Client e cioè, il *CObexClient.* Le impostazioni del Client sono le stesse per qualsiasi applicazione OBEX. La principale differenza è data da come la classe *TObexBluetoothProtocolInfo* viene impostata. La maggior parte delle informazioni, si ottengono in fase di ricerca del device, logicamente sempre con l'utilizzo dello standard Bluetooth e questo accade anche su applicazioni a basso livello *RFCOMM* di cui il protocollo OBEX, come detto prima, si avva-

le. Per motivi di spazio editoriali, abbiamo dovuto purtroppo tralasciare la spiegazione su come cercare un device con connessione Bluetooth attiva (il *Device Discovery*), potrete però avvalervi degli esempi forniti con l'SDK della Nokia e soprattutto con l'esempio che stiamo utilizzando, dove è contenuto tutto il codice necessario per il *Device Discovery.* La costruzione del *CObexClient,* in riferimento al protocollo Bluetooth, avviene mediante la funzione *ConnectToServerL* ed è reperibile nel sorgente *ObjectExchangeClient.cpp.*
La su citata funzione, è parte della classe *CObjectExchangeClient,* come possiamo osservare nella linea di codice seguente:

```
void CobjectExchangeClient::ConnectToServerL()
{
```

IMPOSTIAMO IL CLIENT

Il seguente codice invece si occuperà della creazione e del settaggio del Client, che dovrà effettuare la connessione al Server:

```
  TObexBluetoothProtocolInfo protocolInfo;
protocolInfo.iTransport.Copy(KServerTransportName);
protocolInfo.iAddr.SetBTAddr(iServiceSearcher->
                                        BTDevAddr());
protocolInfo.iAddr.SetPort(iServiceSearcher->Port());
```

Una volta impostato e creato il Client, dovremo connetterlo:

```
  if (iClient)
    {  delete iClient;
       iClient = NULL; }
  iClient = CObexClient::NewL(protocolInfo);
  iClient->Connect(iStatus);
  SetActive();
```

INVIARE E RICEVERE UN OBEX OBJECT

Per far si che il Client invii un oggetto OBEX, dobbiamo logicamente crearlo. L'esempio ci viene in aiuto anche in questo, mediante la funzione

```
CObjectExchangeClient::ConstructL:
void CObjectExchangeClient::ConstructL()
{
   ...
```

La trasmissione dell'oggetto OBEX, consiste in due parti principali: Un set di Headers e una serie di dati veri e propri, detti *PayLoad.* Creando poi l'oggetto *CObexNullObject,* fa in modo che il sistema operativo invii soltanto l'insieme di informazioni di intestazione. Nel nostro caso, l'informazione di intestazione includerà il solo messaggio che l'applicazione Client deve inviare al Server, quindi il famosissimo messaggio *"Hello World".*

```
  iCurrObject = CObexNullObject::NewL();
  iCurrObject->SetNameL(L("Hello World"));
}
```

Creato l'oggetto, l'invio del messaggio avviene mediante la chiamata alla funzione *Put():*

```
void CObjectExchangeClient::SendMessageL()
{  if (iState != EWaitingToSend)
    {   User::Leave(KErrDisconnected);  }
  else if (IsActive())
    {  User::Leave(KErrInUse); }
   iClient->Put(*iCurrObject, iStatus);
   SetActive();
}
```

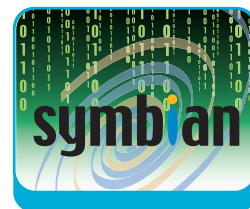
Per la ricezione, non viene fatto altro che salvare il Log di ricezione dati. Questo si può ottenere sapendo principalmente che quando il *CObexServer* riceve le informazioni, avvisa l'applicazione Client tramite le funzioni, come spiegato all'inizio di questo articolo, dette *Observer* e definite nella classe *MObexServerNotify.* Quando l'oggetto e quindi i dati sono stati ricevuti, la funzione *PutCompleteIndication* è innescata. In questa funzione sono immagazzinati il nome ed i dati che il Server estrapolerà:

```
TInt CObjectExchangeServer::PutCompleteIndication()
{  iLog.LogL(iObexBufObject->Name());
   return KErrNone;
}
```

Una volta ricevuto il tutto, potremo provvedere allo spegnimento del server, per l'applicazione Server e del client per l'applicazione Client, logicamente se non avremo bisogno più né del Server, né del Client. Questo potremo farlo solo quando avremo disconnesso il Client dal Server tramite la funzione *Disconnect();* della classe *CObexClient.* Successivamente, a sconnessione avvenuta, potremo anche cancellare l'oggetto *CObexClient.*

CONCLUSIONI

Abbiamo visto quanto l'utilizzo del protocollo OBEX in ambiente Symbian, sia utile e soprattutto produttivo. Questo protocollo ci permetterà di creare applicazioni che possano interagire con Devices di qualsiasi genere, a patto che supportino lo standard Bluetooth o l'iRDA o il trasferimento dati via seriale.

Marcello Scala

Bluetooth
Protocollo di trasferimento ad onde radio.

OBEX Protocol
Object Exchange Protocol.

PayLoad
Riferimento di un pacchetto, frame o altra unità per la trasmissione dati.

RFCOMM
Serial Port Emulation

IMMAGINI ANNOTATE CON JAVA

L'ANNOTAZIONE DI UN'IMMAGINE CONSISTE NELL'EVIDENZIARE RIQUADRI DELLA STESSA A CUI ASSOCIARE UNA BREVE DESCRIZIONE. QUESTO ARTICOLO ILLUSTRA COME CREARE UN COMPONENTE SWING DOTATO DI QUESTA FUNZIONALITÀ.

Lo scopo di questo articolo è creare un'applicazione che ci consenta di modificare delle foto, applicandovi dei segni di evidenziazione che ne mettano in rilievo particolari aree. Ad ogni riquadro di evidenziazione sarà possibile associare una descrizione. In **Figura 1** è presente un esempio: sull'immagine di un tramonto cittadino sono stati evidenziati tre punti. Da sinistra, un traliccio, delle case, ed una parte di cielo illuminato in modo particolare. Ovviamente, la presenza di una etichetta ne riduce la leggibilità da parte dell'utente, anche se è possibile scegliere livelli diversi di trasparenza. Si noti che il ritaglio è evidenziato da un riquadro giallo trasparente, in modo da non coprire la parte di immagine relativa, che invece traspare. L'annotazione è invece scritta in bianco, utilizzando un font di dimensioni ridotte. Tutti questi aspetti grafici, con la relativa realizzazione, saranno descritti in seguito.

Fig.1: Il programma in azione visualizza una immagine annotata

COME FUNZIONERÀ IL PROGRAMMA?

Quando l'applicazione verrà avviata, l'immagine sarà priva di annotazioni e di qualunque segno di evidenziazione. Cliccando sull'immagine sarà possibile disegnare un riquadro che contrassegnerà la parte evidenziata. Inizialmente il riquadro sarà di colore bianco trasparente. Una volta delimitato un riquadro sarà possibile premere il tasto *INVIO*; questo evento, intercettato dal software, comporta la presentazione di una finestra di dialogo per la digitazione della descrizione da associare al ritaglio. Una volta confermata la descrizione, il ritaglio viene aggiunto all'immagine.

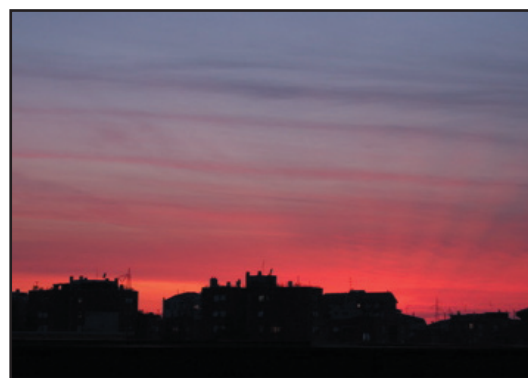



Fig.2: All'inizio dell'esecuzione non sono presenti annotazioni

SIAMO PRONTI PER INIZIARE

Vediamo come realizzare queste funzionalità partendo dalla classe principale dell'applicazione, quella che contiene il metodo *main()*. La parte di codice più interessante riguarderà il costruttore:

- per prima cosa verrà caricata l'immagine di prova, contenuta nel file *DSC_0080.jpg*, utilizzando le *API Image IO;*
- poi verrà creato il pannello che gestice le immagini annotate;
- verrà creato un *JTabbedPane*, utile per inserire un gestore di eventi di accesso alla tastiera. Questo elemento serve ad intercettare il tasto *INVIO*, per invocare la funzionalità di aggiunta di una annotazione. Questo gestore di evento è di tipo *KeyListener.*
- Il pannello *AnnotableImagePanel* viene inserito

nel *JTabbedPane*, che a sua volta viene inserito in una finestra di tipo *JFrame*, che viene poi visualizzata.

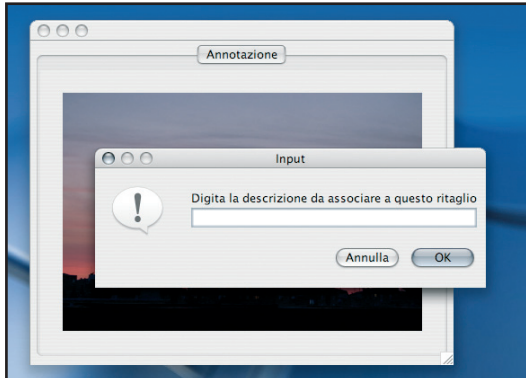



Fig.3: *L'applicazione in esecuzione*

Il codice sorgente completo della classe è il seguente:

```
/**
 * Main
 */
package it.edmaster.ioprogrammo.annotazione;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.net.URL;
import javax.imageio.ImageIO;
import javax.swing.JFrame;
import javax.swing.JTabbedPane;
public class Main {
   public Main() throws IOException {
      //carica l'immagine di prova
      URL url = getClass().getResource ("/DSC_0080.jpg");
      BufferedImage image = ImageIO.read(url);
      //crea il pannello di annotazione immagini
      final AnnotableImagePanel annotableImage
               Panel = new AnnotableImagePanel(image);
      //inserisce l'immagine all'interno di un
      //pannello su cui viene inserita la gestione
      //della pressionedei tasti
      JTabbedPane pane = new JTabbedPane();
      pane.add( annotableImagePanel, "Annotazione" );
      pane.addKeyListener( new KeyAdapter() {
         public void keyPressed(KeyEvent e) {
            if (e.getKeyCode() == KeyEvent.VK_ENTER) {
               e.consume();
               annotable ImagePanel.editClip();
            }
         }
      } );
      //crea una finestra a cui associa il pannello
      //creato al passo precedente
      JFrame frame = new JFrame();
      frame.getContentPane().add( pane );
      frame.pack();
```

```
      frame.setVisible(true);
   }
   /**
    * @param args
    * @throws IOException
    */
   public static void main(String[] args) throws
                                             IOException {
      new Main();
   }
}
```

GRAPHICS O GRAPHICS2D

L'oggetto *Graphics* appartiene alle API *AWT* e rappresenta un generico contesto grafico su cui è possibile disegnare testo, linee, cerchi, riquadri ed altro.

Con l'introduzione di Java2, è stato aggiunto l'oggetto *Graphics2D*, che supporta funzionalità più potenti per la grafica bidimensionale.

IL PANNELLO IMMAGINI

La classe di base è dunque abbastanza banale. Infatti, tutte le funzionalità sono raccolte nella classe *AnnotableImagePanel*. Vediamone il codice:

```
/**
 * AnnotableImagePanel
 */
package it.edmaster.ioprogrammo.annotazione;
import java.awt.AlphaComposite;
import java.awt.Color;
import java.awt.Component;
import java.awt.Composite;
import java.awt.Font;
import java.awt.FontMetrics;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Image;
import java.awt.Point;
import java.awt.Rectangle;
import java.awt.Shape;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.MouseMotionListener;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import javax.swing.BorderFactory;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.SwingUtilities;
/**
 * Pannello che implementa immagini annotabili.
                                  Ciascuna annotazione
```

GLOSSARIO

LA CLASSE COMPOSITE

Composite **è una classe di AWT che permette di definire numerose funzionalità di fusione tra il tratto di disegno e lo sfondo. La classe concreta** ***AlphaComposite*** **agisce sul canale alfa (trasparenza). Un valore alfa di 1.0 significa** ***"completamente opaco"*****. I valori possibili sono inclusi tra l'intervallo 0.0 e 1.0.**

```
 * è identificata da un riquadro, a cui è associata
                                        una descrizione.
 * Ciascun riquadro è rappresentato da un oggetto
                          di tipo <code>Clipping</code>.
 *
 * @author max
 */
public class AnnotableImagePanel extends JPanel {
  /** immagine da visualizzare */
  private Image image;
  /** dimensione bordo interno */
  private final int borderSize = 20;
  /** etichetta utilizzata per visualizzare
                                        l'immagine */
  private JLabel internalLabel;
  /** punto iniziale di trascinamento */
  private Point initialPoint;
  /** indica che è in corso un'operazione di
                                   trascinamento */
  private boolean dragging = false;
  /** punto finale di trascinamento */
  private Point endPoint;
  /** elenco dei ritagli associati all'immagine */
  private List clippings = new ArrayList();
  /** oggetto Composite utilizzato per disegnare
                            riquadri in trasparenza */
  private Composite composite =
AlphaComposite.getInstance(
                        AlphaComposite.SRC_OVER, 0.2f);
```

Il costruttore inizializza il pannello e chiama il metodo *createUI()*, per creare gli elementi dell'interfaccia utente:

```
  /**
   * Crea un pannello annotabile
   * @param image
  */
  public AnnotableImagePanel(Image image)
  {
     super();
     this.image = image;
     //ritaglio di prova
     //clippings.add(Clipping.create(297, 147, 97, 31,
     //"raggi di luce che vengono prodotti dal
     //tramonto inquinato della città di Bollate"));
     createUI();
  }
```

Nel metodo *createUI()* per prima cosa viene creato un oggetto *ImageIcon*, per cui viene creata una classe anonima che ridefinisce il metodo *paintIcon()*. Nella ridefinizione, viene invocato il metodo *drawClippings()*, che si occupa di visualizzare i ritagli associati all'immagine. Questo comporta che ad ogni visualizzazione di questa immagine vengano ridisegnati anche i ritagli:

```
 /**
 * crea l'interfaccia utente
 */
 private void createUI()
 {
   ImageIcon imageIcon = new ImageIcon(image)
   {
      public synchronized void paintIcon( Component
                arg0, Graphics arg1, int arg2, int arg3)
      {
         super.paintIcon(arg0, arg1, arg2, arg3);
         drawClippings();
      }
   };
```

L'*ImageIcon* così creata viene utilizzata per creare una JLabel, un modo standard in Swing per visualizzare una immagine a video. L'etichetta viene poi inserita nel pannello e viene impostato un bordo, per rendere più gradevole l'aspetto del pannello:

```
 internalLabel = new JLabel( imageIcon );
 add( internalLabel );
 setBorder(
   BorderFactory.createEmptyBorder(
      borderSize, borderSize,
         borderSize, borderSize)
 );
```

A questo punto è necessario creare alcuni gestori di evento, in modo da poter intercettare i movimenti del mouse e gestire così la definizione dei ritagli. Il primo gestore è *MouseMotionListener*, che intercetta l'evento *mouseDragged()*. Il funzionamento è il seguente: se non è in corso un'operazione di *dragging* (il valore del flag omonimo è *false*), la attiva, memorizzando il punto iniziale da cui è partito il mouse. Poi imposta il punto finale, e chiama il metodo *trackMove()*, che visualizza il riquadro individuato da questi due punti. L'evento *mouseClicked()* permette di intercettare clic del pulsante, che indica la volontà dell'utente di cancellare un ritaglio attivo. L'evento *mouseReleased()* indica invece il rilascio di un pulsante, il che indica la fine di un'operazione di disegno del ritaglio:

```
 //traccia il movimento del mouse salvando il punto
```

NOTA

INFORMAZIONI SUI FONT

L'oggetto *FontMetrics* permette di conoscere tutti i dettagli numerici di un particolare font. La funzionalità più banale che viene offerta è quella di calcolare l'occupazione di spazio di una stringa, se disegnata con un particolare font, ma questa classe dispone di numerosi altri metodi per conoscere tutte le caratteristiche dimensionali di ciascun carattere del font.

IL METODO SWINGUTILITIES.INVOKELATER()

Il codice presentato nell'articolo ha fatto uso del metodo *SwingUtilities.InvokeLater()*, permette di invocare un blocco di codice, formalizzato come implementazione dell'interfaccia *Runnable*, solo al termine dell'esaurimento di tutti gli eventi nella coda *AWT*. In questo modo si è sicuri che il proprio codice venga eseguito solo al termine di tutte le operazioni di aggiornamento dell'interfaccia utente.

```
//iniziale di trascinamento
addMouseMotionListener(new MouseMotionListener() {
  public void mouseDragged(MouseEvent e)
  {
    if (!dragging)
    {
      initialPoint = e.getPoint();
      dragging = true;
    }
    endPoint = e.getPoint();
    trackMove();
    e.consume();
  }
  public void mouseMoved(MouseEvent e) { }
  } );
  addMouseListener( new MouseAdapter()
  {
    public void mouseClicked(MouseEvent e)
    {
      deleteMove();
      e.consume();
    }
    public void mouseReleased(MouseEvent e)
    {
      dragging = false;
      e.consume();
    }
  } );
}
```

CREARE UN RITAGLIO

I metodi coinvolti nella definizione di un nuovo ritaglio sono *trackMove()* e *editClip()*. Il primo disegna un riquadro a partire dal punto iniziale fino a quello finale, ottenuto estraendo le coordinate dai campi *initialPoint* e *endPoint*. Poi viene estratto il contesto grafico del componente che contiene l'immagine, viene impostato il colore bianco e viene disegnato il bordo del ritaglio. Poi viene impostato il *Composite* che indica al sistema di disegnare con una trasparenza del 20% e viene riempito il ritaglio:

```
/**
 * disegna il ritaglio in fase di definizione
 *
 */
private void trackMove()
{
  //ridisegna l'immagine
  internalLabel.repaint();
  final Rectangle b = internalLabel.getBounds();
  SwingUtilities.invokeLater( new Runnable()
  {
    public void run()
    {
      int x = (int)(initialPoint.getX() - b.getX());
      int y = (int)(initialPoint.getY() - b.getY());
      int width=(int)(endPoint.getX() - b.getX()) - x;
      int height=(int)(endPoint.getY() - b.getY()) - y;
      Graphics2D g = (Graphics2D
                          )internalLabel.getGraphics();
      g.setColor(Color.WHITE);
      g.drawRect(x, y, width, height);
      g.setComposite( composite );
      g.fillRect(x+1, y+1, width-1, height-1);
    }
  } );
}
```

Una volta che l'utente ha selezionato un ritaglio, può aggiungerlo all'immagine. Questa operazione viene avviata dalla pressione del tasto *INVIO*, che viene intercettato dal *KeyListener* impostato sul *JTabbedPane*. Questo invoca il metodo *editClip()*, che presenta una finestra di dialogo in cui l'utente deve digitare la descrizione da associare al ritaglio. Una volta inserita viene creato un oggetto *Clipping*, che rappresenta un singolo ritaglio. Questo oggetto contiene le coordinate del riquadro da evidenziare e la descrizione da associare.
Dispone inoltre di metodi statici *create()* che vengono utilizzati per creare oggetti di questo tipo.
Al termine dell'operazione il ritaglio attivo viene cancellato:

```
/**
 * presenta la finestra per la digitazione
 * della descrizione del ritaglio
 */
public void editClip()
{
  Point p1 = initialPoint;
  Point p2 = endPoint;
  String m = JOptionPane.showInputDialog(
             this, "Digita la descrizione da associare
                              a questo ritaglio");
  if (m != null && m.length() > 0)
  {
    //crea un ritaglio e lo aggiunge all'elenco
```

NOTA

CARICARE UN'IMMAGINE

Il modo più facile, a partire da Java 1.4, per caricare una immagine in memoria è utilizzare la classe *ImageIO*, invocando il metodo *read()*:

```
BufferedImage image =
    ImageIO.read(new File(
        "fotografia.jpg" ));
```

L'oggetto ritornato è di tipo *BufferedImage*, che descrive un tipo particolare di immagine il cui buffer dati è in qualche modo accessibile.

CONVERSIONI DI TIPO

La classe *ImageIO* si può utilizzare anche per effettuare conversioni di tipo. Ad esempio, per cambiare il formato di una immagine in formato *jpeg* nel formato *png*. In questo caso è sufficiente caricare l'immagine e poi salvarla con il metodo *write()*, specificando il formato richiesto:

```
BufferedImage image = ImageIO.read(
        new File("forografia.jpg" ));
File outputFile = new File(
                   "fotografia.png");
ImageIO.write(
        image, "png", outputFile );
```

Attenzione: la piattaforma Java supporta solo *GIF, JPEG, PNG, BMP* e pochi altri formati.

```
        Clipping c = Clipping.create( p1, p2, m );
        clippings.add(c);
        repaint();
    }
    initialPoint = null;
}
```

DISEGNARE I RITAGLI

Proseguendo nella discussione della classe *AnnotableImagePanel*, verranno affrontati ora i metodi di disegno dei ritagli. Il metodo *drawClippings()* si occupa di disegnare tutti i ritagli associati all'immagine, estraendo il contesto *Graphics* dall'etichetta, i suoi limiti *(bounds)* e chiamando il metodo *drawClippings():*

```
private void drawClippings() {
  if (internalLabel != null) {
     Graphics2D g = (Graphics2D)
                         internalLabel.getGraphics();
     Rectangle b = internalLabel.getBounds();
     drawClippings(g, b);
  }
}
```

Il metodo effettivo di disegno è il seguente. Come si nota dal codice, viene impostata una iterazione sulla lista dei ritagli dell'immagine.
L'impostazione del codice è simile al metodo *trackMove()*, con la differenza che il colore del ritaglio è giallo, e non bianco, come la selezione attiva. Inoltre il ritaglio ha una descrizione, diversamente dalla selezione.
I passaggi svolti dal metodo *draw Clippings()* sono i seguenti:

- determina le coordinate del ritaglio;
- disegna il bordo;
- disegna il contenuto (con effetto trasparenza), salvando il precedente oggetto *Composite;*
- crea un font più piccolo di quello di default per disegnare le descrizioni;
- imposta il colore bianco ed il *Composite* originale;
- salva il ritaglio *Swing* attivo dell'immagine;
- imposta il font e disegna il testo;
- ripristina il vecchio ritaglio.

Il codice è il seguente:

```
/**
 * disegna sull'immagine i ritagli evidenziati
 * @param g
 */
private void drawClippings(Graphics2D g, Rectangle b)
{
  if (g != null && b != null && clippings != null)
  {
    for( Iterator it = clippings.iterator();
    it.hasNext(); )
  {
      Clipping c = (Clipping)it.next();
      Rectangle r = c.getBounds();
      //determina le coordinate del ritaglio
     int x = (int)(r.getX() - b.getX());
     int y = (int)(r.getY() - b.getY());
      int width = (int)r.getWidth();
      int height = (int)r.getHeight();
      //disegna il bordo del riquadro
      g.setColor( Color.YELLOW );
      g.drawRect(x, y, width, height);
      //disegna il contenuto del riquadro,
      //utilizzando la trasparenza
      Composite oldComposite = g.getComposite();
      g.setComposite( composite );
      g.fillRect(x+1, y+1, width-1, height-1);
      //crea un font più picccolo
      //per le descrizioni
      Font font = g.getFont();
      font = font.deriveFont(8f);
      //imposta il colore bianco ed il
      //composite di default
      g.setColor( Color.WHITE );
      g.setComposite( oldComposite );
      //imposta la regione di clipping sullo
      //spazio individuato dal ritaglio
      Shape oldClip = g.getClip();
      g.setClip(x+1, y+1, width-1, height-1);
      //imposta il font da utilizzare e disegna
      //la descrizione
      g.setFont(font);
      drawMultiLineString(g, c.getDescription(), x, y,
                                       width, height);
      //ripristina il vecchio ritaglio
      g.setClip(old Clip);
    }
  }
}
```

Le operazioni di salvataggio dei vecchi *Composite* e *Clip* dell'immagine sono indispensabili, in quanto il disegno dell'annotazione successiva sarebbe intralciata dalle impostazioni di ritaglio e trasparenza impostate per l'annotazione precedente.
Un altro aspetto degno di nota è il metodo *drawMultiLineString()*, creato appositamente per disegnare testo su più linee. La logica di funzionamento di questo metodo è semplice: calcola la dimensione di ciascun carattere, fino a che non viene raggiunta la dimensione massima in pixel del ritaglio corrente. A questo punto il metodo disegna i caratteri fino ad ora individuati e poi va

NOTA

IL PACKAGE SWING

Un altro modo facile per caricare una immagine in Java è utilizzare la classe ImageIcon, presente nel package Swing. Per esempio:

```
ImageIcon ic = new
ImageIcon("icona.jpeg");
```

Per estrarre l'immagine è poi possibile utilizzare il metodo *getImage():*

```
Image image =
        ic.getImage();
```

a capo, ripetendo l'operazione fino a che ci sono caratteri nella stringa, oppure fino a che non viene esaurito lo spazio disponibile in verticale:

```
/**
  * disegna una stringa su più righe
  * @param g
  * @param text
  * @param x
  * @param y
  * @param width
  * @param height
  */
  private void drawMultiLineString( Graphics g,
            String text, int x, int y, int width, int height )
  {
      FontMetrics fm = g.getFontMetrics();
      char[] chars = text.toCharArray();
      int i = 0;
      int start = 0;
      int line = 1;
      int lineWidth = 0;
      //bordo di due pixel
      int border = 3;
      width -= border * 2 + 2;
      //cicla per tutti i caratteri della stringa
      while (i < chars.length)
      {
          int cw = fm.charWidth ( chars[i] );
          lineWidth += cw;
          if (lineWidth >= width || i == chars.length-1)
          {
              //disegna fino al carattere i-1
              int limit = i-1-start;
              //se sono a fine stringa disegno tutto
              if (i == chars.length-1)
              {
                limit = i-start+1;
              }
             //verifico di non superare l'altezza
             //massima
            int dy = (line*fm.getAscent());
            if (border+dy > height)
            {
                break;
            }
            //crea la sottostringa e la scrive
            String s = new String( chars, start, limit);
            g.drawString(s, x+border, y+border+dy);
            start = i-1;
            line++;
            lineWidth = 0;
        }
        i++;
    }
}
```

Ovviamente questo metodo non supporta la sillabazione e la giustificazione del testo: queste funzionalità lo avrebbero reso decisamente più complesso. Un approccio alternativo sarebbe stato utilizzare una JLabel per ciascun ritaglio, utilizzando solamente componenti *Swing* ad alto livello, evitando del tutto l'oggetto *Graphics* e le sue funzionalità. La classe si conclude con il metodo per cancellare il ritaglio in corso:

```
        /**
         * cancella la selezione attiva
         */

        private void deleteMove()
        {
                initialPoint = null;
                endPoint = null;
                internalLabel.repaint();
        }
}
```

CONCLUSIONI

In questo articolo sono state illustrate alcune tecniche per operare con la grafica a basso livello in Java, utilizzando sempre come punto di partenza componenti visuali *Swing*.
Particolarmente interessanti sono le funzionalità per disegnare elementi grafici utilizzando la trasparenza e l'oggetto *FontMetrics*, che permette di conoscere l'occupazione esatta in pixel di caratteri e stringhe, funzionalità che ha consentito di realizzare un semplice disegno di testo multilinea.

Massimiliano Bigatti

QUALE FORMATO

I formati grafici supportati da Java sono limitati.
Per conoscere quali *codec* sono presenti nel sistema è possibile utilizzare sempre la classe *ImageIO,* che dispone di alcuni metodi per conoscere i formati supportati, sia in lettura che in scrittura.
L'esempio di codice seguente ne stampa l'elenco utilizzando un metodo di supporto per stampare un vettore:

```
String[] readers =
    ImageIO.getReaderFormatNames();
print( "Lettori (informali)", readers );

String[] mimeReaders =
       ImageIO.getReaderMIMETypes();
print( "Lettori (MIME)", mimeReaders );
String[] writers =
    ImageIO.getWriterFormatNames();
print( "Scrittori (informali)", writers );
String[] mimeWriters =
       ImageIO.getWriterMIMETypes();
print( "Scrittori (MIME)", mimeWriters );
private void print(
      String header, String[] elements)
{
  System.out.println( header );
  System.out.println( "========" );

  for(int i=0; i<elements.length; i++)
  {
    System.out.println( elements[i] );
  }

  System.out.println( "========" );
}
```

IO PROGRAMMO BY EXAMPLE

IMPARA A PROGRAMMARE IN MODO PRATICO E DIVERTENTE, CON GLI ESEMPI PASSO PASSO CHE TI GUIDANO ALLA COSTRUZIONE DEL CODICE

VUOI INVIARE UN ESEMPIO?

Se sei un programmatore esperto ed hai risolto un problema, puoi aiutare gli altri pubblicando il tuo codice. Proponi i tuoi esempi scrivendo a ioprogrammo@edmaster.it

Come contattarci?

Alla nostra redazione arriva spesso un considerevole numero di email riguardante temi specifici. Per consentirci di rispondere velocemente e in modo adeguato alle vostre domande abbiamo elaborato una FAQ – Frequently Ask Question – o risposte alle domande frequenti in italiano, di modo che possiate indirizzare le vostre richieste in modo mirato.

Problemi sugli abbonamenti

Se la tua domanda ha a che fare con una delle seguenti:

- Vorrei abbonarmi alla rivista, che devo fare?
- Sono un abbonato e non ho ricevuto la rivista, a chi devo rivolgermi?
- Sono abbonato ma la posta non mi consegna regolarmente la rivista, a chi devo rivolgermi?

Contatta **abbonamenti@edmaster.it** specificando che sei interessato a ioProgrammo. Lascia il tuo indirizzo email e indica il numero dal quale vorresti far partire l'abbonamento. Verrai contattato al più presto. Oppure puoi chiamare lo **02 831212**

Problemi sugli allegati

Se riscontri un problema del tipo:

- Ho acquistato ioProgrammo ed il Cdrom al suo interno non funziona. Chi me lo sostituisce?
- Ho acquistato ioProgrammo ma non ho trovato il cd/dvd all'interno, come posso ottenerlo?
- Vorrei avere alcuni arretrati di ioProgrammo come faccio?

Contatta **servizioclienti@edmaster.it**
Non dimenticare di specificare il numero di copertina di ioProgrammo e la versione: con libro o senza libro. Oppure telefona allo **02 831212**

Assistenza tecnica

Se il tuo problema è un problema di programmazione del tipo:

- Come faccio a mandare una mail da PHP?
- Come si instanzia una variabile in c++?
- Come faccio a creare una pagina ASP.NET

o un qualunque altro tipo di problema relativo a tecniche di programmazione, esplicita la tua domanda sul nostro forum: **http://forum.ioprogrammo.it**, uno dei nostri esperti ti risponderà. Le domande più interessanti saranno anche pubblicate in questa rubrica.

Problemi sul codice all'interno del CD

Se la tua domanda è la seguente

- Non ho trovato il codice relativo all'articolo all'interno del cd

Consulta la nostra sezione download all'indirizzo **http://cdrom.ioprogrammo.it**, nei rari casi in cui il codice collegato ad un articolo non sia presente nel cdrom, senza dubbio verrà reso disponibile sul nostro sito.

Idee e suggerimenti

Se sei un programmatore esperto e vuoi proporti come articolista per ioProgrammo, oppure se hai suggerimenti su come migliorare la rivista, se vuoi inviarci un trucco suggerendolo per la rubrica tips & tricks invia una email a **ioprogrammo@edmaster.it**

COME POSSO REALIZZARE UN WEB SERVICE?

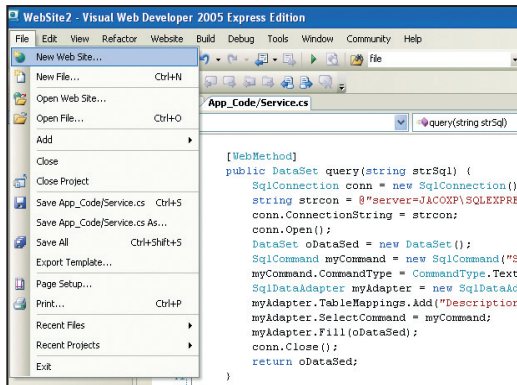
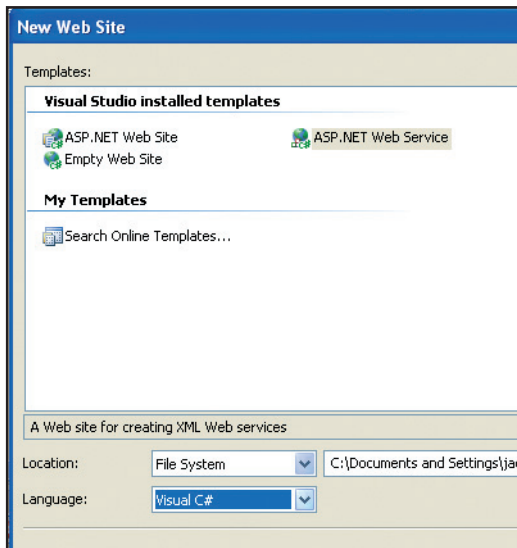
CON VISUAL STUDIO 2005 CI VENGONO IN AIUTO ALCUNI WIZARD MOLTO UTILI, CON PHP VEDREMO CHE LE DIFFICOLTÀ SONO MAGGIORI SOLO IN FASE DI TEST E DEPLOY

C#

VISUAL BASIC.NET

PHP

FACCIAMOLO IN ASP.NET CON C#

1 Iniziamo con il creare un nuovo sito tramite il menu File/New Web Site.

2 Dalla dialog box che segue scegliamo "Web Services" come tipologia del progetto e più in basso selezioniamo il linguaggio C#

3 Aggiungiamo il seguente spezzone di codice

```
[WebMethod]
    public DataSet query(string strSql)
    {
        SqlConnection conn = new SqlConnection();
        string strcon = @"server=
                JACOXP\SQLEXPRESS;Initial Catalog=
            ioProg_DVD_4SQL;User ID=sa; PWD=sa";
        conn.ConnectionString = strcon;
```

```
        conn.Open();
        DataSet oDataSed = new DataSet();
        SqlCommand myCommand = new
            SqlCommand("SELECT * FROM Description
                where Applicazione Like '%" + strSql
                                + "%'", conn);
        myCommand.CommandType =
                            CommandType.Text;
        SqlDataAdapter myAdapter =
                        new SqlDataAdapter();
        myAdapter.TableMappings.Add(
                            "Table", "Table0");
        myAdapter.SelectCommand = myCommand;
        myAdapter.Fill(oDataSed);
        conn.Close();
        return oDataSed;
    }
```

4 Compiliamo il tutto e copiamo il risultante file asmx sul webserver dove vogliamo che il nostro servizio venga esposto.

COME FUNZIONA?

Lo starter kit ci mette a disposizione lo scheletro di un web service che non fa altro che restituire al client la stringa helloworld. Dal codice si capisce subito che per esporre un metodo bisogna farlo precedere dall'etichetta [WebMethod] è esattamente quello che abbiamo fatto noi con il metodo "query".

Il nostro metodo si connette ad un database sql server 2005 e invia a questo una query composta da una parte fissa e da una parte da aggiungere all'operator like. La parte variabile deve essere fornita al web service dal client. Ad esempio se il client fornisse al web service il parametro "ioProgrammo" verrebbero selezionate tutte le righe della tabella description che al loro interno hanno la parola "ioProgrammo".

FACCIAMO IN ASP.NET CON VISUAL BASIC

I passi da eseguire nei punti uno e due rimangono identici a quelli utilizzati per il progetto in linguaggio C#. Il codice da utilizzare invece è il seguente:

```
<WebMethod()> _
    Public Function query(ByVal strSql As String)
```

```
                                        As DataSet
        Dim conn As SqlConnection =
                                New SqlConnection()
        Dim strcon As String =
                "server=JACOXP\SQLEXPRESS;Initial
                Catalog=ioProg_DVD_4SQL;User ID=sa;
                                        PWD=sa"
        conn.ConnectionString = strcon
        conn.Open()
        Dim oDataSed As DataSet = New DataSet()
        Dim myCommand As SqlCommand = New
            SqlCommand("SELECT * FROM Description
            where Applicazione Like '%" & strSql &
                                        "%'", conn)
        myCommand.CommandType =
                                CommandType.Text
        Dim myAdapter As SqlDataAdapter =
                                New SqlDataAdapter()
        myAdapter.TableMappings.Add(
                        "Description", "Table0")
        myAdapter.SelectCommand = myCommand
        myAdapter.Fill(oDataSed)
        conn.Close()
        Return oDataSed

    End Function
```

FACCIAMOLO IN PHP

1 Creiamo un file index.php all'interno di una sottodirectory del web server e come prima operazione sviluppiamo la funzione che si interfaccerà al database

```
function execQuery($strSql)
{
    $connessione = mysql_connect(
                "localhost", "root", "password"
                ) or die("Connessione non
                riuscita: " . mysql_error());
    mysql_select_db("ioprogrammo") or die(
            "Selezione del database non riuscita");
```

COSA È NUSOAP?

Nusoap è una libreria opensource direttamente scaricabile all'indirizzo http://dietrich.ganx4.com/nusoap/ e che implementa un framework intermedio per l'accesso alle funzioni soap e di conseguenza per creare e usare web services in PHP. Non solo si tratta di una libreria affidabile, ma è praticamente l'unica che consente di generare automaticamente il wsdl. In PHP5 esistono una serie di funzioni interne per utilizzare i web services che risultano essere molto più veloci di quelle implementate in Nusoap, tuttavia rimangono ancora sufficientemente limitate per certi aspetti. Di fatto non consentono la generazione automatica del wsdl per tale motivo anche per chi usa PHP5 è consigliabile ancora adesso utilizzare Nusoap. In PHP4 il problema non si pone, di fatto non ci sono funzioni interne a PHP per interfacciarsi con i web services.

```
    /* Esecuzione di una query SQL */
    $query = "SELECT * FROM file where
                Applicazione like '%$strSql%'";
    $risultato = mysql_query($query) or die(
                "Query fallita: " . mysql_error() );

            while ($riga = mysql_fetch_array(
                        $risultato, MYSQL_ASSOC))
    {
    $columns[] = $riga;
    }

    /* Chiusura della connessione */
    mysql_close($connessione);
        //return $columns[0];
    return new soapval('return','tns:Idiot',$columns);
}
```

2 Includiamo il file *nusoap.php*

```
require_once('nusoap.php');
```

3 Instanziamo il server e aggiungiamo un tipo che ci servirà per descrivere i valori di ritorno che devono essere contenuti nello schema XML Soap

```
$server = new soap_server();
// Initialize WSDL support
$ns="http://localhost/~fabio/stock/services.php";
$server->configureWSDL('services', $ns);
$server->wsdl->schemaTargetNamespace=$ns;

$server->wsdl->addComplexType("MyType",
                        "complexType",
                        "array",
                        "all",
                        "",
array( "ID" =>
                array("ID" => "ID",
                        "type" => "xsd:string"),
                "Applicazione" => array(
                "Applicazione" => "Applicazione",
                        "type" => "xsd:string"),
                "File" => array("File" =>
                "File", "type" => "xsd:string"),
            "Sottotitolo" => array(
            "Sottotitolo" => "Sottotitolo", "type"
                        => "xsd:string"),
                "Descrizione" => array(
                "Descrizione" => "Descrizione",
                        "type" => "xsd:string"),
                    "NumeroRivista" => array(
            "NumeroRivista" => "NumeroRivista",
                        "type" => "xsd:string"),
                    "IDmc" => array("IDmc"
```

```
=> "IDmc", "type" => "xsd:string"),
"Dim" => array("Dim" =>
"Dim", "type" => "xsd:string"),
"posiz" => array(
"posiz" => "posiz", "type" => "xsd:string"),
"Link" => array("Link" =>
"Link", "type" => "xsd:string"),
"Evidenza" => array(
"Evidenza" => "Evidenza", "type" =>
"xsd:string"), "Indispensabili" =>
array("Indispensabili" => "Indispensabili",
"type" => "xsd:string")
)

);
```

4 Inseriamo le informazioni relative al protocollo di trasporto e agli altri dati che serviranno per la descrizione del WSDL

```
$server->register('execQuery', // method name
    array('strSql' => 'xsd:string'), // input parameters
    array('return' => 'tns:Idiot'),
                                        // output parameters
    'urn:services', // namespace
    'urn:services#execQuery', // soapaction
    'rpc', // style
    'encoded', // use
    'Ritorna un array contenente i valori richiesti'
                                        // documentation
);
```

5 Infine invochiamo il web services

```
$HTTP_RAW_POST_DATA = isset(
                        $HTTP_RAW_POST_DATA) ?
                        $HTTP_RAW_POST_DATA : '';
$server->service($HTTP_RAW_POST_DATA);
```

COME POSSO CONTROLLARE LO STATO DEL TASTO CAPS LOCK?

Per controllare il tasto CAPS LOCK il framework .NET mette a disposizione il metodo IsKeyLocked.
Facciamone un esempio in Visual Basic .NET

```
If Control.IsKeyLocked Then
    MessageBox.Show("CAPS LOCK Inserito",
        "", MessageBoxButtons.OK, _
        MessageBoxIcon.Exclamation)
Else
    MessageBox.Show("CAPS LOCK Non
        Inserito", "", MessageBoxButtons.OK, _
        MessageBoxIcon.Exclamation)
EndIf
```

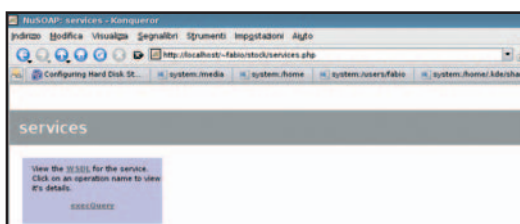
COME POSSO VERIFICARE CHE UN WEB SERVICE FUNZIONI?

NORMALMENTE È POSSIBILE CONNETTERSI DIRETTAMENTE ALL'INDIRIZZO DEL WS ED OTTENERE INFORMAZIONI SUL WSDL, L'OUTPUT PERÒ DIPENDE MOLTO DA COME IL SERVIZIO È STATO REALIZZATO

FACCIAMOLO CON PHP & NUSOAP

PHP & NUSOAP

.NET

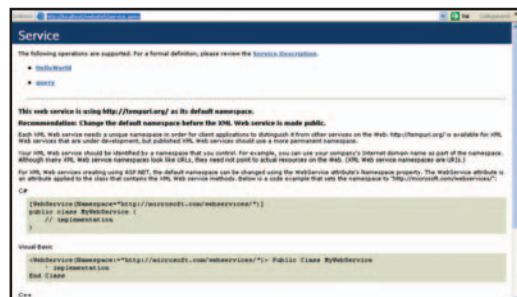
1 Aprite un normale browser e connettetevi al servizio Web. Ad esempio *http://localhost/~vostrahome/servizio/nomeservizio.php*. Supposto che il Web Service sia stato realizzato con PHP e Nusoap l'output sarà il seguente:

La pagina web contiene un link al file WSDL e uno per uno i link ai metodi esposti dal webservices

2 Clicchiamo sul link al *wsdl* e otteniamo il file wsdl in formato chiaramente leggibile. Si noti

che non usando SOAP sarebbe piuttosto complicato generare a mano il file in questione

3 Clicchiamo su uno dei metodi esposti dal webservice ed otterremo un riferimento alla documentazione relativa a quel particolare metodo. Si notino in particolare i parametri in input ed output relativi al metodo in questione

FACCIAMOLO CON .NET

1 Anche i in questo caso è necessario collegarsi alla url che espone il web service. Nel nostro esempio *http://localhost/website5/service.asmx.*

Esattamente come nel caso di PHP verrà mostrata una pagina che contiene i link ai metodi implementati dal servizio

2 Per ottenere la descrizione contenuta nel file WSDL questa volta è necessario utilizzare il link "service description"

3 Cliccando sul link che riporta il nome di uno dei metodi si ottiene questa volta non solo una pagina con la descrizione dei parametri accettati in input e in output, ma anche una o più caselline di input che ci consentono di testare il servizio al di la di un normale client

COME POSSO INSTALLARE UN WEB SERVICE IN IIS?

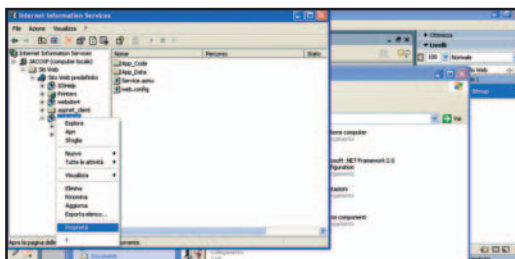
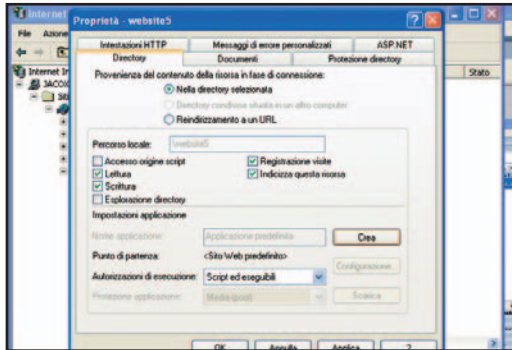
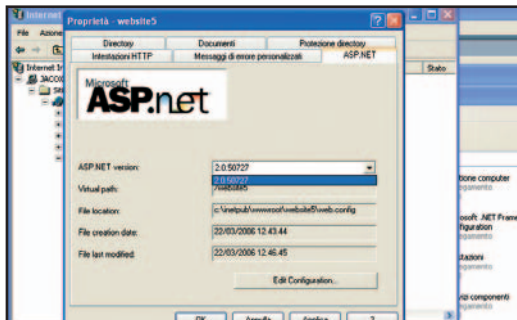
ANCHE IN QUESTO CASO È SUFFICIENTE COPIARE I FILE CHE LO COMPONGONO NELLA DIRECTORY CHE OSPITERÀ IL SERVIZIO. TUTTAVIA IN IIS LA DIRECTORY IN QUESTIONE DEVE ESSERE CONFIGURATA COME APPLICAZIONE. PER FAR QUESTO SI PUÒ PROCEDERE IN DUE MODI

COME SI INSTALLA UN WEB SERVICE IN APACHE?

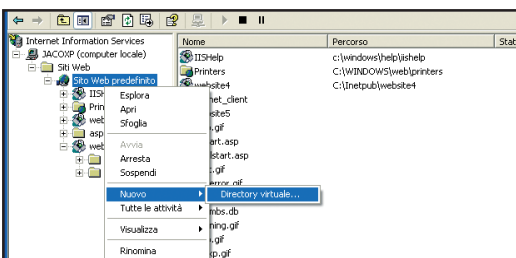
È sufficiente copiare il file *php* e il realtivo *nusoap.php* nella directory che esporrà il servizio.

1 Creare una directory al di sotto della *wwwroot* di IIS.

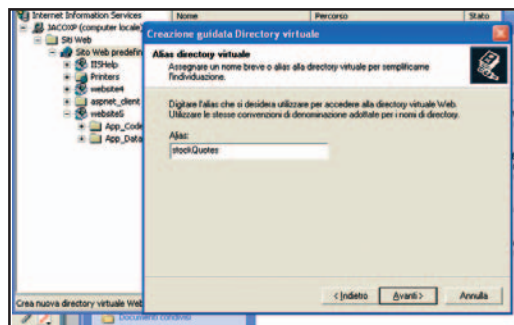
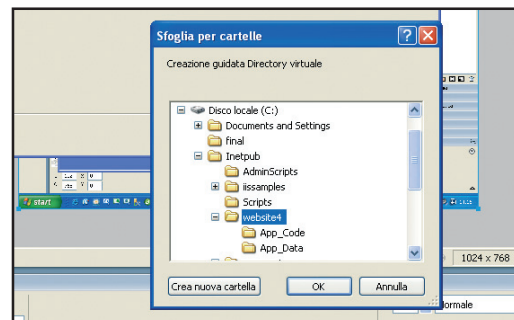
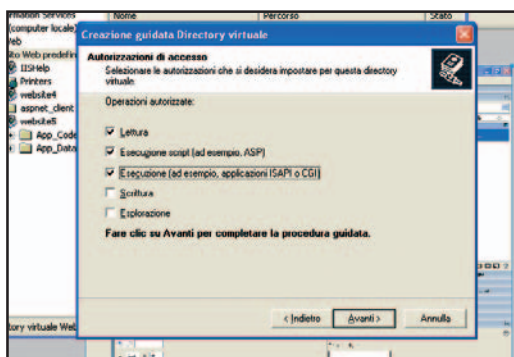
2 Portarsi in pannello di controllo/strumenti di amministrazione e avviare *Internet Information Services.*

3 Nella finestra che compare selezionare la directory in cui installare il web services e cliccare con il tasto destro del mouse, di seguito sulla voce proprietà.

4 Nella finestra delle proprietà cliccare su "crea applicazione"

5 Portatevi nella la tabsheet "Asp.net" e nella casella a discesa che riguarda la versione di Asp.net installata selezionate quella relativa al framework con cui avete sviluppato il Web Services

A questo punto potete applicare tutte le modifiche e copiare i file nella directory opportuna.

FACCIAMOLO CON UNA DIRECTORY VIRTUALE

1 La differenza con il metodo precedentemente descritto consiste nel fatto che in questo caso la directory che conterrà i file sarà posizionata al di fuori della wwwroot. In questo caso portatevi nel web site principale, cliccate con il tasto destro e selezionate "nuova directory virtuale"

2 Si avvierà una procedura guidata. Al secondo punto vi verrà richiesto di inserire un alias per la vostra directory virtuale. Utilizzate un'etichetta significativa ad esempio il nome del vostro progetto: stockQuotes

E SE IIS RESTITUISCE ERRORE?

Se avete installato IIS dopo avere installato Visual Studio Express Edition, è probabile che non siano stati creati gli utenti corretti per potere eseguire le applicazioni ASP.NET in Windows. Per risolvere il problema portatevi nella directory di installazione del framework. Tipicamente *C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727* e digitate *aspnet_regiis -i*. Ovviamente sostituite al percorso indicato quello relativo alla vostra installazione. Verranno automaticamente creati gli utenti necessari a far funzionare IIS con ASP.NET 2.0 e il problema precedente sarà risolto

3 Nel passo successivo vi verrà chiesto di selezionare la directory fisica a cui la vostra directory virtuale è associata. Cercatela nell'albero delle directory tramite il tasto sfoglia

4 Impostate i permessi per la nuova directory, selezionate almeno il permesso d'esecuzione

5 La procedura termina qui. Potete settare l'installazione di ASP.NET ripartendo dallo step 5 del metodo precedentemente

E SE NON HO IIS?

Potete ugualmente provare il funzionamento del vostro Web Services se avete utilizzato Visual Studio 2005 Express. Il prodotto di Microsoft dispone infatti di un web server integrato. È sufficiente cliccare sul bottoncino "play" nella barra degli strumenti di Visual Web Developer e verrà lanciato un nuovo Web Server su una porta random della vostra macchina

RICAVARE INFORMAZIONI SUI TIPI

Per conoscere a runtime l'oggetto *Type* per un particolare tipo è possibile ricorrere in C# all'operatore *typeof*:

```
Type t=typeof(string);
//restituisce System.String
```

In Vb.NET l'operatore equivalente è *GetType*:

```
Dim t as Type
t=GetType(string)
```

Se si vuole lo stesso risultato, ma per conoscere il tipo di un'istanza o di un'espressione, è invece utilizzabile il metodo *GetType*, ereditato dalla classe *Object*:

```
Type t=obj.GetType();
```

Oppure il metodo statico *Type.GetType*, con il nome completo sotto forma di stringa, che restituisce *null (nothing in vb.net)* se non riesce a risolvere il nome del tipo:

```
t = Type.GetType("System.Int32")
```

Esiste in Vb.NET un operatore *Typeof*, da non confondere con il typeof di C#, e che si usa congiuntamente all'operatore is, formando un'espressione del tipo *TypeOf expr is type*:

```
if(TypeOf i is Integer)
 ...
End if
```

L'operazione restituisce *true* se l'espressione è compatibile con il tipo specificato. Ad esempio sarebbe vera per una *Form* testandola con il tipo *Control*.

```
If (TypeOf form1 Is Control) Then
Console.Write("form1 è un Control")
End If
```

Per effettuare un test sulla compatibilità dei tipi in C# invece si usa il solo operatore *is*:

```
if(obj is Control)
{
//obj è compatibile con Control }
```

COME POSSO UTILIZZARE UN WEB SERVICES?

I DIVERSI METODI DIPENDONO DAI LINGUAGGI, IN TUTTI I CASI LA BASE RIMANE SEMPRE SOAP

PHP

VISUAL BASIC.NET

C#

FACCIAMOLO CON PHP

1 Prima di tutto includiamo la libreria nusoap.php che contiene le varie classi che ci consentono di lavorare in modo semplificato con il webservices

```
<?php
require_once('nusoap.php');
```

2 Creiamo un istanza del client, e tramite il file wsdl informiamo l'istanza di quali sono i metodi che il WS espone e della sua descrizione. Ovviamente il parametro passato al client corrisponde all'indirizzo del WS. Nel nostro caso il wsdl viene generato automaticamente dallo script, in altri potremmo avere bisogno di referenziare direttamente il file wsdl

```
$client = new soapclient_n("http://localhost
                /~fabio/stock/services.php?wsdl", true);
```

3 Controlliamo se ci sono eventuali errori

```
$err = $client->getError();
if ($err) {
    echo '<h2>Errore ne costruttore</h2><pre>'
                                       . $err . '</pre>';
}
```

4 Richiamiamo il metodo esposto dal webservices passandogli eventuali parametri sotto forma di array

```
$result2 = $client->call('execQuery', array(
                                   'strSql' => 'test'));
```

5 controlliamo eventuali errori e stampiamoli eventualmente a video

```
if ($client->fault) {
    echo '<h2>Fault</h2><pre>';
    echo '</pre>';
}
else {
    $err = $client->getError();
    if ($err) {
        echo '<h2>Error</h2><pre>' . $err
                                    . '</pre>';
```

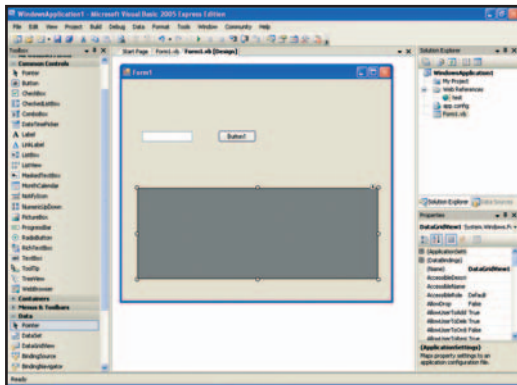
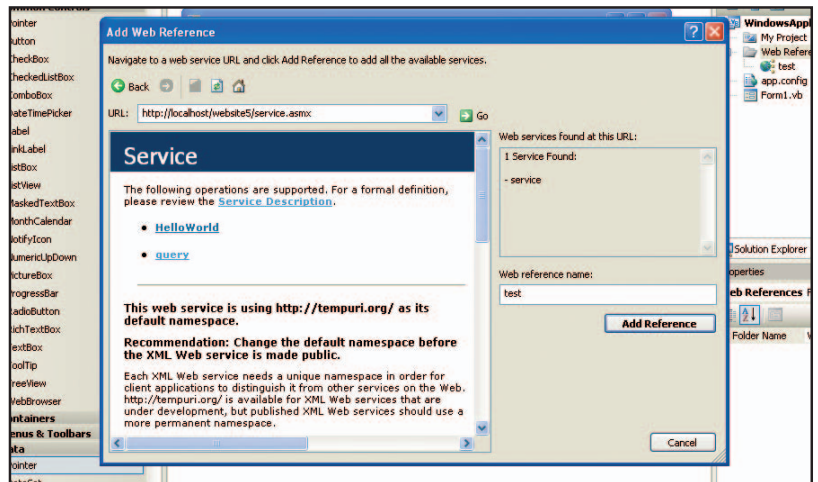
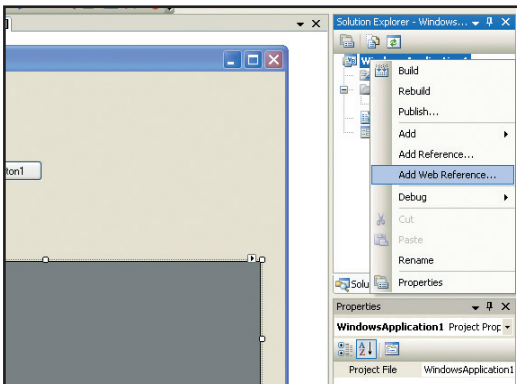
6 Se non ci sono errori stampiamo il risultato a video

```
} else {
        // Display the result
        echo '<h2>Result</h2><pre>';
        print_r($result2);
    echo '</pre>';
    }
}
```

7 Se vogliamo possiamo stampare anche il debug delle comunicazioni avvenute fra client e server

```
echo '<h2>Request</h2>';
echo '<pre>' . htmlspecialchars(
        $client->request, ENT_QUOTES) . '</pre>';
echo '<h2>Response</h2>';
echo '<pre>' . htmlspecialchars(
        $client->response, ENT_QUOTES) . '</pre>';
// Display the debug messages
echo '<h2>Debug</h2>';
echo '<pre>' . htmlspecialchars(
        $client->debug_str, ENT_QUOTES) . '</pre>';
?>
```

FACCIAMOLO IN VISUAL BASIC.NET

1 Su una form trasciniamo una DataGridView, una TextBox e un bottone

2 Clicchiamo nel solution explorer con il tasto destro sul nome del progetto, nel menu a tendina che ne risulta selezioniamo "Add Web Reference"

3 Nella dialog box che compare, indichiamo l'url che espone il web services. Pochi secondi dopo ci comparirà la descrizione del wsdl e dei metodi esposti. Clicchiamo su Add Reference. Possiamo se vogliamo, cambiare il nome che comparirà come etichetta del WS

4 Clicchiamo due volte sul bottone per gestire il codice relativo all'evento OnClick. Il codice da inserire è il seguente

```
Private Sub Button1_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                                Handles Button1.Click
    Dim ws As New test.Service
    Dim ds As DataSet = ws.query(
                        TextBox1.Text.ToString())
    'DataGridView1.DataSource = ds
    DataGridView1.DataSource = ds.Tables("Table0")
    'DataGridView1.DataMember = "Table0"
    DataGridView1.Enabled = True
    DataGridView1.Refresh()
  End Sub
End Class
```

FACCIAMOLO IN C#

1 Fino al punto 4 il procedimento rispecchia fedelmente quello utilizzato da Visual Basic. Il codice di gestione diventa invece il seguente

```
private void button1_Click(object sender, EventArgs e)
{
   test.Service ws = new test.Service();
   DataSet ds = new DataSet();
   ds = ws.query(textBox1.Text.ToString());
   // DataGridView1.DataSource = ds
   dataGridView1.DataSource = ds.Tables["Table0"];
   // DataGridView1.DataMember = "Table0"
   dataGridView1.Enabled = true;
   dataGridView1.Refresh();
}
```

CHE COSA VUOL DIRE WSDL?

NEI NOSTRI ESEMPI SUI WEB SERVICES CI SIAMO SPESSO RIFERITI A UN FILE WSDL COME QUELL'ELEMENTO CHE CONTIENE LA DESCRIZIONE DEL SERVIZIO. MA COSA È ESATTAMENTE E COME FUNZIONA?

I Web Services non sono il primo tentativo di creare un ponte di comunicazione fra elementi installati in luoghi geograficamente distanti, tuttavia hanno apportato una notevole dose di cambiamento proprio per la loro caratteristica di aggirare i limiti imposti dai linguaggi per diventare "Intercomunicanti" in modo del tutto indipendente dalla piattaforma di sviluppo e dal sistema operativo. L'idea è semplice. Lo sviluppatore A crea un programma (un web service) e lo rende disponibile su un sito web. Lo sviluppatore B interroga quel sito web e senza avere conoscenze di come è implementato il web services che c'é sotto ne usa le informazioni. Ad esempio il Web Service potrebbe restituire l'elenco delle città italiane con più di 50.000 abitanti. Al programmatore B non importa sapere con quale linguaggio sia stato sviluppato il Web Service o su che sistema operativo giri, importa solo il risultato. Per ottenere il risultato deve sapere solo quali metodi del web service invocare e niente altro. È proprio a questo che serve WSDL, si tratta di un formato standard per la descrizione formale di un Web Services. Ogni Web Service deve esporre il proprio file WSDL. Per la difficoltà di implementazione del formato, tipicamente il programmatore non scrive a mano il suo file WSDL ma si affida a framework come nusoap o asp.net ad esempio, che consentono la generazione dinamica del file WSDL. In questo modo interrogando il web service con un certo parametro, tipicamente ?WSDL si ottiene la sua descrizione formale, generata automaticamente dal framework utilizzato per lo sviluppo. Se volete potete provare ad analizzare un file WSDL.

```
<message name=
            "execQueryRequest">
<part name="strSql" type=
                "xsd:string"/>

</message>
   <message name=
           "execQueryResponse">
<part name="return" type=
                  "tns:MyType"/>
</message>
```

in questo caso viene indicato che il WS deve ricevere una stringa in ingresso e restituisce un output in formato MyType. Più avanti si può trovare la definizione di MyType

```
<xsd:complexType name=
                    "MyType">
<xsd:all>
<xsd:element name="ID" type=
        "xsd:string" ID="ID"/>
<xsd:element name=
  "Applicazione" type="xsd:string"
        ID="ID" Applicazione=
               "Applicazione"/>
</xsd:all>
</xsd:complexType>
```

QUALI SONO I COMANDI BASE DI MYSQL DA LINEA DI COMANDO?

UNA RAPIDA CARRELLATA SUI PRINCIPALI METODI PER CREARE, ELIMINARE I DB, E IMPORTARE ED ESPORTARE DATI IN FORMATO SQL

MYSQL

1 Il comando principale per accedere a mysql è

```
mysql -unomeutente -ppassword database
```

se accedete come root potete evitare di selezionare il database. root è l'utente amministrativo a cui sono concessi i privilegi per eseguire ogni operazione sul db.

2 Per creare un nuovo database potete usare il comando

```
mysqladmin -unomeutente -ppassword create
                               nomedatabase
```

3 Per creare un nuovo utente che abbia accesso al database appena creato entrate in mysql come utente root e poi digitate

```
grant all privileges on database.* to utente
                   identified by 'password';
```

Potete modificare il simbolo "*" con l'unica tabella a cui intendete concedere il permesso, e modificare "all" con i privilegi che ritenete più opportuni

4 Per eliminare un database potete usare il comando

```
mysqladmin -unomeutente -ppassword drop database
```

5 Per esportare un database in formato SQL

```
mysqldump -unomeutente -ppassword database
                                  > nomefile
```

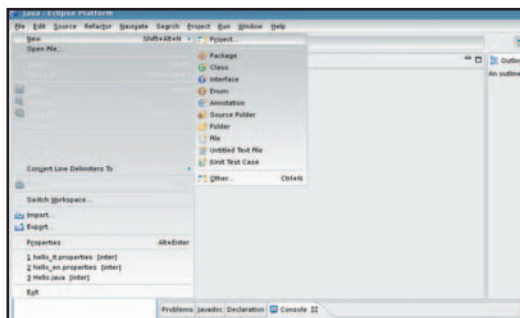
6 Per importare un database dal formato mysql

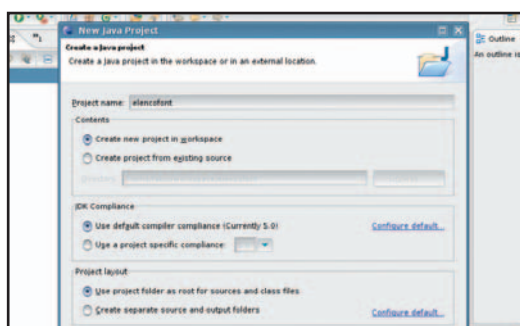
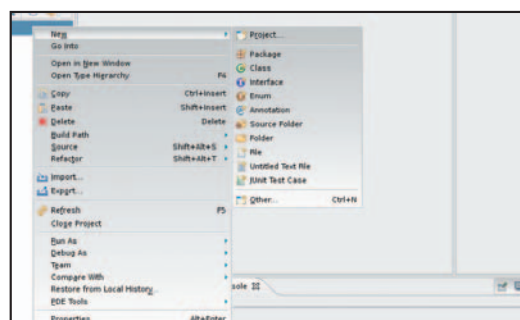
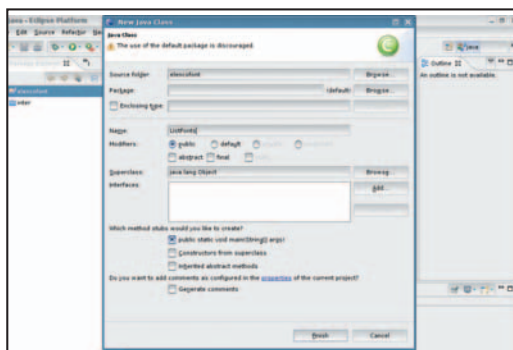
```
mysql -unomeutente -ppassword database < nomefile
```

COME ELENCARE I FONT DISPONIBILI?

SFRUTTIAMO ALCUNE CLASSI DI JAVA PER AVERE INFORMAZIONI SUI FONT INSTALLATI NEL SISTEMA

JAVA

1 Apriamo eclipse e scegliamo File/New Project

2 Dalla dialog box successiva scegliamo “Java Project”

3 Diamo un nome significativo al progetto, ad esempio “elencofont” e clicchiamo su finish

4 Aggiungiamo una nuova classe agendo sul pulsante sinistro del mouse cliccando sul nome del progetto e scegliendo “New Class”

5 Diamo un nome significativo alla classe ad esempio “ListFonts” e scegliamo di fare generare anche il main tramite l'apposito switch

6 Aggiungiamo in testa alla classe gli import che ci serviranno per recuperare le nostre informazioni

```
import java.awt.GraphicsEnvironment;
```

7 E modifichiamo il main in modo opportuno

```
public static void main(String[] args) {
    ListFonts lf = new ListFonts();
    lf.printFonts(); }
```

8 Non ci resta che scrivere il metodo printFonts

```
public void printFonts() {
    GraphicsEnvironment ge = GraphicsEnvironment.
                    getLocalGraphicsEnvironment();
    String[] fonts = ge.getAvailableFontFamilyNames();
    System.out.println("Numero di Fonts
                    disponibili: " + fonts.length);
    System.out.println("Elenco dei Fonts disponibili:");
    for(int i=0; i<fonts.length; i++) {
        System.out.println("\t" + fonts[i]);
    }
}
```

COME FUNZIONA?

Tutto è demandato al metodo printFonts che instanzia un nuovo oggetto di classe GraphicsEnvironment e utilizza il metodo getAvailableFontFamilyNames() per riempire un array di stringhe che viene poi stampato con un classico ciclo di for.

ANTEPRIMA SUL NUOVO JAVA 6 SE

MIGLIORATA L'INTEGRAZIONE CON IL DESKTOP DEL SISTEMA OPERATIVO DI RIFERIMENTO: È POSSIBILE IMPLEMENTARE IL COMPORTAMNETO DELLA BARRA DI STATO E GESTIRE L'AGGIUNTA DI ICONE CON MENU POPUP PERSONALIZZABILI

Gli sviluppatori di mezzo mondo non avevano ancora digerito per bene la versione 5 della Standard Edition di Java, che già ci si trova a fare i conti con la 6, coincidente con la 1.6.0 developer version. In realtà si tratta di una beta, nome in codice Mustang, ma la curiosità è tale e tanta da stuzzicare anche il developer più annoiato. Partendo con l'esplorazione della "beta six", non possiamo non apprezzare una considerevole attenzione verso l'universo desktop. Ciò è rilevabile in primis dall'interfaccia utente e dall'integrazione con il sistema operativo di riferimento, e questa, a ben vedere, è una novità a dir poco rivoluzionaria, considerata la proverbiale ritrosia dei progettisti di casa Sun nel preoccuparsi di contaminazioni costruttive con l'ambiente circostante, probabilmente per una attenzione quasi maniacale nonché esclusiva verso le proprie creazioni e una devozione esagerata verso la filosofia dell'essere piuttosto che per quella dell'apparire. I meriti di questo cambiamento sono da ascrivere anzitutto al progetto JDesktop Integration Components (JDIC), che offre alle applicazioni Java l'accesso alle caratteristiche rilevabili dal desktop del sistema operativo in cui l'ambiente di sviluppo stesso si trova ad essere ospitato: questa operazione di incoraggiamento verso la collaborazione con le applicazioni desktop attive e maggiormente utilizzate, dal browser al client di posta, alla gestione risorse, alle utility per la stampa o la riproduzione multimediale, sarà sicuramente vista di buon grado da parte dell'utenza, oltre a rappresentare un ottimo viatico per migliorare l'indice di gradimento verso Java ad opera dei più diffidenti. Non a caso, tra le new entry visibili con maggiore evidenza, c'è da segnalare la presenza di una fantastica API nuova di zecca: nientepopodimenoché il package *java.awt* .Desktop, quello che, grazie ad appositi action events garantirà alle nuove applicazioni realizzate in codice Java la piena integrazione con le desktop applications più comuni per la creazione e la modifica di file nei formati più popolari. Proseguendo, segnaliamo poi il supporto degli splash screen, con la possibilità che questi informino l'utente circa l'avvio di un'applicazione. Da segnalare anche i miglioramenti nelle Java Foundation Classes e nella gestione degli oggetti Swing, con maggiori integrazioni che semplificano il layout complessivo delle applicazioni e una personalizzazione avanzata, udite udite, del drag and drop. Insomma, se pensiamo ai burrascosi rapporti con Windows, pare proprio che fra un po' la cosa potrà ritenersi acqua passata. Java sembra infatti viaggiare verso una integrazione sempre più spinta con il sistema Microsoft, che la porterà fra non molto ad essere considerata, ci si passi il paragone, non più lo scomodo inquilino che crea sempre problemi, ma quello dal carattere un po' sui generis con cui però si può andare comunque d'accordo.

DOVE SCARICARE LA JAVA SE 6 BETA

All'indirizzo http://java.sun.com/javase/6/download.jsp possiamo effettuare il download gratuito della Java SE Development Kit (JDK) 6 Beta. Disponibile anche la *Runtime Environment (JRE) 6 Beta* che consente agli utenti finali di eseguire applicazioni Java.

GESTIAMO LA BARRA DI STATO

Due presenze interessanti si trovano nel package java.awt: si tratta in particolare delle classi SystemTray e TrayIcon, il cui compito è relativo alla gestione della barra di stato che, in Windows come in altri ambienti, consente di raccogliere riferimenti a file, servizi ed applicazioni in attività: ebbene, attraverso la gestione di azioni ed eventi, con un'applicazione Java è possibile manipolare con il mouse gli oggetti che fanno parte della barra o che ruotano attorno ad essa, implementando operazioni di aggiunta, rimozione e modifica della caratteristiche relative a ciascuna voce. In particolare, un'istanza della classe SystemTray, propria per ciascuna applicazione, può contenere una o più istanze di oggetti TrayIcons, ciascuna definita per mezzo di un'immagine, un menu popup ed un insieme di listener associati. Ciascuna istanza di un oggetto TrayIcons potrà essere aggiunta alla barra (o meglio, all'istanza che la definisce in astrazione Java) attraverso il metodo add(java.awt .TrayIcon), così come potrà essere rimossa per mezzo del metodo speculare remove(java.awt.TrayIcon). L'implementazione in codice impone che ciascuna applicazione Java presenti una singola istanza SystemTray che consenta all'applicazione stessa di interfacciarsi con la barra

interfacciarsi con la barra di stato del sistema per tutto il tempo in cui l'applicazione è attiva. L'istanza SystemTray può essere ottenuta attraverso il metodo getSystemTray(), mentre un'applicazione non potrà creare la propria istanza della classe SystemTray.
L'accesso alla barra di stato del sistema avviene attraverso una serie di istruzioni del tipo:

```
TrayIcon iconaBarra = null;
if (SystemTray.isSupported()) {
  // si apre l'if: la barra è supportata
   // restituisci l'istanza della barra
   SystemTray barra =
                    SystemTray.getSystemTray();
   // adesso carichiamo un'immagine
   Image immagine =
           Toolkit.getDefaultToolkit.getImage(...);
   // ora creiamo un action listener per metterci
  // in ascolto dell'azione di default eseguita
  // sull'icona della barra
   ActionListener listener = new ActionListener() {
       public void actionPerformed(ActionEvent e) {
          // esegui l'azione di default dell'applicazione
          // ...}
   };
```

Se invece vogliamo popolare la barra con un menu popup, ed aggiungere icone personalizzate, scriveremo:

```
// creiamo un menu popup
PopupMenu popup = new PopupMenu();
// creiamo una voce di menu per l'azione di default
MenuItem defaultItem = new MenuItem(...);
defaultItem.addActionListener(listener);
popup.add(defaultItem);
/// ... aggiungiamo altre voci costruiamo
// una TrayIcon
trayIcon = new TrayIcon(image, "Esempio ", popup);
// impostiamo le proprietà corrette
trayIcon.addActionListener(listener);
// aggiungiamo l'icona alla barra
try {
        tray.add(trayIcon);
} catch (AWTException e) {
        System.err.println(e);}
} // si chiude l'if
           else {
        // la barra non è supportata: disabilitiamo
        // l'opzione barra nell'applicazione
        // o eseguiamo altre azioni
        ...
}
```

Possiamo addirittura gestire il cambiamento dello stato dell'applicazione

```
  // se lo stato dell'applicazione è cambiato
```

```
  // aggiorniamo l'immagine
  if (trayIcon != null) {
      trayIcon.setImage(updatedImage); }
```

GENERAZIONE... INTERATTIVA!

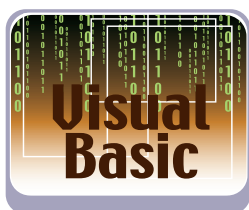
Proseguendo, l'altro grande vantaggio della Standard Edition 6 è legato al core, ossia a tutto ciò che ruota attorno al linguaggio e agli strumenti adoperati per gestirne le potenzialità. Guardiamo per esempio al compilatore: la nuova API Java Compiler garantisce che il codice sorgente Java sia compilato all'interno di un'applicazione. Durante la compilazione, l'applicazione ha accesso al contenuto informativo elaborato in relazione alle librerie, compreso di eventuali warning, errori e altri messaggi che potrebbero essere generati. A ben vedere, questa caratteristica esprime un vantaggio da non sottovalutare: se infatti si prova a costruire lo strato di accesso ai dati di un'applicazione d0i cui si stava effettuando il build, il codice scritto genererà e compilerà le classi adoperate per accedere alle tabelle di database dell'applicazione. Alla fine, avremo un file JAR che sarà generato, buildato e deployato come parte degli script Ant del sistema. E il fatto che le classi siano state compilate all'interno dell'applicazione fa della generazione del codice un processo interattivo: è possibile modificare e costruire le classi in maniera iterativa… pare poco? Per abilitare il Java scripting, la Standard Edition 6 supporta il framework JSR 223, che offre l'accesso del linguaggio di scripting alle porzioni più nascoste del linguaggio. In tal modo, è possibile localizzare gli engine di scripting e richiamarli per eseguire script a runtime. L'API Scripting consente di offrire il supporto Java per il linguaggio di scripting scelto, così come il Web Scripting Framework garantirà al codice script la generazione di contenuto Web all'interno di qualsiasi Servlet container. Per effettuare le operazioni di debugging, la piattaforma Java di riferimento (JPDA) è stata migliorata per intercettare eventuali deadlock e generare tracciati di stack per oggetti monitor eventualmente bloccati. Inoltre, Java SE 6 garantisce che, per motivi di carattere diagnostico, gli agent si allaccino ad una JVM in esecuzione. La presenza di un full stack trace nella eccezione java.lang.OutOfMemory e la conseguente generazione di un log di fatal error al riempimento dell'heap contribuisce a migliorare la gestione della memoria, così come, sempre al riempimento dell'heap, una nuova opzione della virtual machine consente di eseguire un particolare script.
Le novità sono ancora molte, avremo occasione di andare in profondità nel momento in cui la versione definitiva sarà rilasciata

Luigi Caputo (luigi@edmaster.it)

WEB SERVICES

La presenza di alcune API fa sì che la nuova Java SE 6 supporti egregiamente i Web services. L'API XML *Digital Signatures* garantisce Web services totalmente Java-based attraverso l'implementazione di operazioni di crittografia su dati in formato XML, così come l'API *JAX-WS 2.0* ha aggiornato la libreria *JAX-RPC*. Proseguendo, si segnalano miglioramenti a *Java-XML Binding* (JAXB) *2.0*, compresi il supporto dello schema XML e il *class binding* verso uno schema. Infine, l'API STaX *(Streaming API for XML)* offre un'API bidirezionale allo scopo di leggere e scrivere oggetti XML mediante uno stream di eventi, compresa la capacità di bypassare le sezioni e di concentrarsi invece sulle sottosezioni di documenti.

SCRIVI IN VISUAL BASIC LEGGI IN WORD

COME UTILIZZARE IL CONTROLLO RICH TEXT BOX E INTERAGIRE CON IL FAMOSO FORMATO RTF. UN ESEMPIO IMPORTANTE CHE CI CONDUCE A POTERE STAMPARE UN DOCUMENTO OFFICE PARTENDO DA UNA NOSTRA APPLICAZIONE

Mentre da Microsoft arrivano indiscrezioni sulla prossima versione di Office - Office 2007, noi abbiamo pensato di dedicare un articolo all'interazione tra Visual Basic e Office XP. L'abbiamo fatto, non perché siamo preistorici, come recita una nota pubblicità della Microsoft, ma perché riteniamo ancora interessanti le applicazioni realizzabili in questo ambiente. Infatti, Office XP grazie a Visual Basic ed ai formati Open supportati, come RTF *(Rich Text Format)* e XML, può essere reso più produttivo, in numerosi campi, ed aperto verso altre applicazioni e sistemi. Ricordiamo che l'RTF, come il PDF, è una specifica pubblica per la creazione, lo scambio e la distribuzione di documenti, per cui oltre a dare una sommaria descrizione del formato RTF, vedremo come utilizzarlo nell'ambiente Visual Basic – Office XP. In particolare introdurremo il controllo *RichTextBox* (casella di testo RTF) che utilizzeremo per realizzare un raccoglitore di documenti Office (Word ed Excel). Poi presenteremo un'altra applicazione che permette di creare documenti Word in base ad un Modello, cioè un file *.dot*.

RICHTEXTBOX, RTF E OLE

Il controllo *RichTextBox*, che supporta il formato RTF, permette di manipolare testo, immagini e oggetti *OLE* in modo avanzato rispetto ai tradizionali *TextBox:* per capirci, esso è assimilabile ad un raccoglitore di oggetti. La pagina delle proprietà del controllo, tra l'altro, permette di specificare il percorso di un file da caricare, l'aspetto del controllo (*multiline, scrollbars*), abilitare o no i menu popup (*AutoVerbMenu*), ecc. Il controllo *RichTextBox* inoltre permette di incorporare al suo interno oggetti *OLE* e di aprire e salvare file sia in formato RTF che Ascii. Per esempio nel controllo è possibile inserire tabelle Excel, file Word, Grafici ecc, anche attraverso il Drag & Drop. Il controllo *OLE* (che si trova tra i controlli primitivi di Visual Basic) permette d'inserire degli oggetti nei Form. Ricordiamo che, la tecnologia *OLE*, per certi versi, è stata soppiantata da quella basata sugli Activex che ormai sono gli elementi standard dell'interfaccia utente. Nei nostri esempi il controllo *OLE* verrà utilizzato per inserire degli oggetti nel controllo *RichTextBox*. Per quanto riguarda il formato RTF, la codifica del testo è "tagged", nel senso che il contenuto del documento è specificato in dei marcatori o Tag.
Un esempio di file RTF è il seguente:

```
{\rtf1\ansi\ansicpg1252\deff0\deflang1040
{\fonttbl{\f0\fnil\fcharset0
Times New Roman;}
      }
\viewkind4\uc1\pard\f0\fs60
{\b io{\i Programmo}
}
\par }
```

Scrivete il codice precedente in un file di testo, salvatelo con estensione RTF ed otterrete il documento che contiene la parola ioProgrammo. Nella guida in linea di Visual Basic è riportato l'elenco delle istruzioni RTF interpretate dal controllo *RichTextBox*. Negli esempi, del controllo *OLE* utilizzeremo le proprietà *OLETypeAllowed* e *Class* e il metodo *InsertObjDlg*. *OLETypeAllowed* imposta il tipo oggetto supportato dal contenitore *OLE*, oggetti collegati, se *=VbOLELinked*, incorporati invece se *=VbOLEEmbedded*. *Class* restituisce il nome di classe di un oggetto incorporabile cioè *Word.Document.x*, *Excel.Sheet.x* (*x* è il numero di versione) ecc. *InsertObjDlg* visualizza la finestra che permette di selezionare l'oggetto da incorporare. Del controllo *RichTextBox* utilizzeremo il metodo *SaveFile*, che consente di salvare il contenuto in formato *RTF* o *TXT*, e la collezione *OLEObjects* che contiene gli oggetti incorporati nel controllo. Per inserire un oggetto nella collezione bisogna utilizzare il metodo *ADD* che ha la seguente sintassi:

Add indice, chiave, documentoorigine, classe

L'unico parametro obbligatorio è il nome del documento da incorporare (documentooriginе). Il valore di *Classe* si può ricavare con l'oggetto *OLE*, la chiave è una stringa univoca che potrà servire per selezionate l'oggetto con il metodo *Item*.

UN RACCOGLITORE OFFICE

In questo paragrafo descriviamo come creare

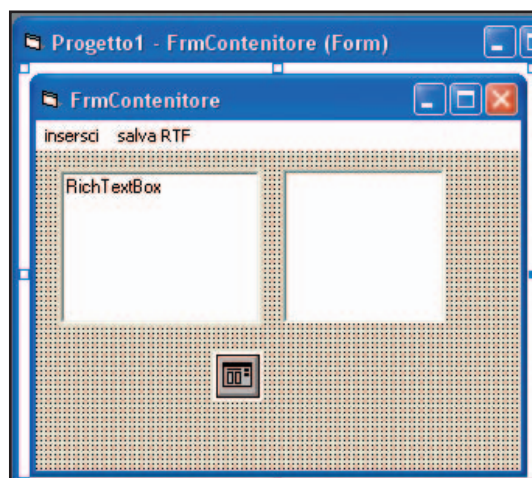



Fig. 1: **La form con oggetto OLE e RichTextBox**

un raccoglitore di documenti Office XP utilizzando un controllo *OLE* ed un *RichTextBox*.

1 Create un nuovo progetto e referenziate la libreria *Microsoft Rich Textbox Control (RICHTX32.ocx)* e la *MS Common Dialog Control.* Sul form disponete un RichTextBox, un controllo *OLE* e un menu a tendina. In quest'ultimo prevedete i menu *Inserisci* e *Salva RTF. Inserisci* permette di selezionare l'oggetto da inserire nel RichTextBox, *Salva RTF* invece permette di salvare il contenuto del RichTextBox in un file in formato RTF.

2 Nella parte dichiarativa della Form bisogna inserire la dichiarazione della variabile num che servirà come contatore degli oggetti inserite nel RichTextBox. Nella *Form_Resize,* invece, bisogna inserire il codice che permette di ridimensionare il RichTextBox in base alle dimensioni del Form. Il controllo *OLE1,* naturalmente, non deve essere visibile.

```
Dim num As Integre

Private Sub Form_Resize()
 ScaleMode = 1
 On Error Resume Next
 testoRTF.Left = 0
```

```
 testoRTF.Height = (Me.Height - testoRTF.Top)
 testoRTF.Width = Me.Width - 100
End Sub
```

3 Il codice da inserire nelle voci di menu è il seguente.

```
Private Sub mnuinserisci_Click()
 OLE1.OLETypeAllowed = vbOLEEmbedded
 OLE1.InsertObjDlg
 If OLE1.Class <> "" Then
  CommonDialog1.FileName = ""
  CommonDialog1.ShowOpen
  If CommonDialog1.FileName <> "" Then
   num = num + 1
   testoRTF.OLEObjects.Add , "oggetto" + _
   CStr(num), CommonDialog1.FileName, OLE1.Class
  End If
 End If
End Sub
```

Nella *mnuinserisci_Click* prima è abilita la maschera del controllo *OLE*, poi è invocato il metodo *ShowOpen* del *CommonDialog1* che permette di selezionare un documento dell'applicazione OLE scelta. Questo documento, grazie alla *ADD*, è inserito, nel RichTextBox, nel punto in cui si trova il cursore del mouse. Il codice per salvare il contenuto del RichTextBox in un file RTF è il seguente.

```
Private Sub mnuRTF_Click()
 CommonDialog1.FileName = ""
 CommonDialog1.Filter = "(*.rtf Documento RTF)|*.rtf"
 CommonDialog1.ShowSave
 testoRTF.SaveFile Me.CommonDialog1.FileName,
                                          rtfRTF
 Exit Sub
errore:
MsgBox Err.Description
Err.Clear

End Sub
```

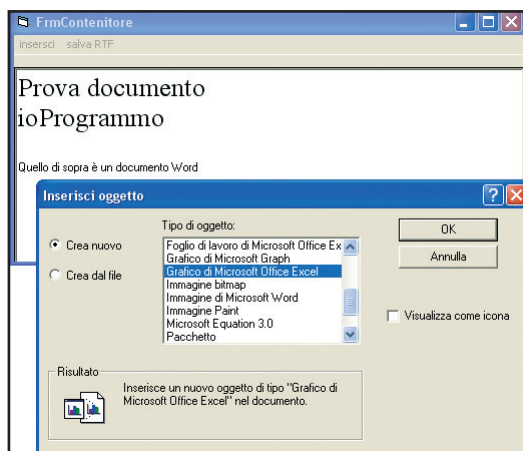
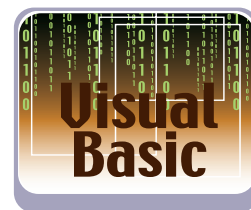
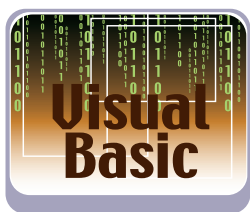



Fig. 2: **La fase di inserimento di un oggetto**

FORMATI APERTI- RTF, PS ECC...

In generale si parla di testi non formattati (nel caso del TXT), testi formattati (HTML, RTF, DOC) e testi formattati per la stampa (PS, PDF). Il formato può essere di tipo proprietario o aperto, in quest'ultimo caso le specifiche di definizione sono di dominio pubblico. Tra i formati (o linguaggi di descrizione) pubblici ricordiamo il PDF, il PS e l'RTF. L'RTF è stato introdotto dalla Microsoft e, per certi versi, ricalca le caratteristiche del formato DOC.

ALCUNI ELEMENTI DI WORD

Descriviamo sommariamente alcuni elementi Word che utilizzeremo negli esempi. I Template o i modelli di Word sono dei file con estensione .dot che contengono gli elementi per la definizione delle caratteristiche di singoli documenti. Il modo più semplice per creare un *Template* è selezionare il menu *"Wiew/Task Pane"* e sulla finestra che compare *(Task Pane)* selezionare *General Templates*. Questa azione fa comparire la finestra New Document con l'elenco dei tipi di documenti che si possono creare, per il nostro scopo bisogna selezionare *Black Document* con l'opzione *Create New Templates*. Il *Template*, e quindi i documenti da esso ricavati, possono essere personalizzati inserendo elementi (oggetti o macro) nelle sue varie parti. Come vedremo, negli esempi, nel nostro *Template* prevediamo tre *Procedure (Macro)* che, interfacciandosi con un'applicazione Visual Basic, permettono d'impostare gli elementi dell'intestazione, del corpo e del piè del documento. La procedura relativa al corpo tra l'altro inserisce due *Fields* di Documento associati alle relative Variabili che, naturalmente, abiliteremo da Visual Basic. Le *Variabili di documento* sono delle variabili che servono per memorizzare dei valori da utilizzare nell'intero documento; le *Variabili di documento* sono membri della collezione *Variables*. Per inserire una *Variabile* nella collezione si può utilizzare il metodo *ADD*. I *Fields* (Campi) *di Documento* di tipo *DOCVARIABLE*, invece, servono per rendere visibili i valori delle *Variabili*. I *Fields*, a loro volta, sono membri della collezione *Fields* che rappresenta l'insieme dei *Fields* associati al documento attivo. Per inserire un *Field* alla collezione bisogna usare il metodo *ADD* che ha la seguente sintassi.

```
Add(Range, Type, Text, PreserveFormatting)
```

dove Range rappresenta l'area del documento in cui i *Fields* operano, nei nostri esempi scriveremo

Selection.Range (dove *Selection* rappresenta la selezione corrente su una finestra o un pannello). *Type*, opzionale, rappresenta il tipo di testo che si vuole associare al *Field*, nel nostro caso sarà *wdFieldEmpty* (campo vuoto) che è il tipo di *Default*. *Text*, anch'esso opzionale, rappresenta il testo d'associare al *Field*, nel nostro caso il testo lo fornisce una variabile di Documento, caricata con il valore passato da Visual Basic. Nella procedura *Wcorpo*, descritta sotto, che gestisce gli elementi del corpo del documento, infatti, è previsto il seguente codice *Text:= "DOCVARIABLE"" VarBody1 """* ect. Infine c'è il parametro *PreserveFormatting* che preserva la formattazione del campo durante l'aggiornamento – *UpDate* – *dei Fields*. L'*UpDate dei Fields* è necessario per assicurare il refresh dei valori visibili nel documento, questo soprattutto dopo che si è assegnato un nuovo valore alla variabile associata al campo, come sarà chiaro tra poco.
Nella procedura, Visual Basic, per invocare le Macro del modello utilizziamo il metodo *Run* di *Word Application*. La sintassi del metodo è la seguente:

```
espressione.Run(MacroName, varg1, …, varg30)
```

Il metodo richiede il nome della Macro nel formato *MacroName:="Nome"* e, se necessari, i parametri nel formato *varg1:="valore del parametro"*. Notate che è possibile passare fino a 30 parametri. Per spostare il cursore tra due *Fields "DocVariable"* utilizziamo il metodo *GoTo*. Esso restituisce un oggetto *Range* che rappresenta la posizione iniziale dell'elemento specificato. La sintassi di *GoTo* è la seguente:

```
Espressione. GoTo(What, Which, Count, Name).
```

What rappresenta il tipo di elemento in corrispondenza del quale verrà spostata la selezione, nel nostro caso è appunto un *Field*. *Which* specifica come verrà fatto lo spostamento, nel nostro caso *wdGoToNext (Successivo)*. *Count* specifica di quanti elementi bisogna spostarsi. *Name* specifica il nome del tipo di elemento, nel nostro caso è *"DOCVARIABLE"*.

APPLICAZIONE VISUAL BASIC - WORD

Di seguito presentiamo l'applicazione Visual Basic che permette di creare dei documenti Word, sulla base di un modello *(Template)*. Questi saranno personalizzati, con immagini, oggetti e stringhe, utilizzando i comandi presenti su un Form Visual Basic. In particolare prevediamo di gestire l'intestazione del documento, il corpo e il piè. All'intestazione passiamo un'immagine (il logo) ed un te-

SMART TAG E SMART DOCUMENT

Gli Smart Tag sono gli elementi intelligenti di Office XP. Essi dal punto di vista dell'utilizzatore sono delle stringhe (label) associate ad un tipo d'informazione, in genere, racchiusa in un'applicazione esterna ai Documenti Office. Dal punto di vista del programmatore, invece, lo Smart Tag è un componente COM che implementa alcune interfacce della libreria Microsoft Smart Tags.
Per attivare gli Smart Tag, nei documenti Word, si deve usare il menu Tolls/AutoCorrect Options … e poi, nella finestra associata al menu, selezionare l'etichetta Smart Tags. In questa etichetta sono mostrati i vari tipi di Smart Tag.
Gli Smart Document, invece, sono stati introdotti con Office 2003. Essi per quanto riguarda l'intelligenza "contestuale" sono l'evoluzione degli Smart Tag. Gli Smart Document sono strumenti che uniscono i vantaggi delle applicazioni desktop con quelli della tecnologia XML.

sto; al corpo passiamo il testo dell'oggetto e quello del contenuto vero e proprio. Nel piè, invece, impostiamo la data e il numero di pagina. Iniziamo presentando gli elementi da inserire sulla Form, poi passeremo a descrivere il modello Word ed infine introdurremo il codice da inserire nella Form.

LA FORM DEL PROGETTO VB

1 Create un nuovo progetto e referenziate le librerie: *Microsoft Rich Textbox Control* e *Common Dialog*. Sulla Form prevedete gli elementi che permettono di definire ed inviare il testo e l'immagine al documento Word. In particolare inserite: 2 TextBox, nominati *txtintestazione* e *txtoggetto;* un RichTextBox nominato *txtcorpo;* una PictureBox, nominata *Picturelogo* che conterrà l'anteprima dell'immagine del logo. Inoltre inserite due CheckBox per specificare che cosa bisogna impostare nel piè pagina cioè *Checkdata* e *CheckPagina*. Infine inserite i pulsanti per creare il documento Word (nominato *Crea*), per inviare i dati (*Invia*), per salvare il documento *(salva)*, per salvare il contenuto del TxtCorpo in formato RTF e per caricare il logo (pulsante *logo*). Prevedete anche i pulsanti conta e controlla per utilizzare i servizi di Word che permettono di contare i caratteri e di verificare la sintassi del testo presente in *TxtCorpo*. Disponete il tutto come in **Figura 3**.

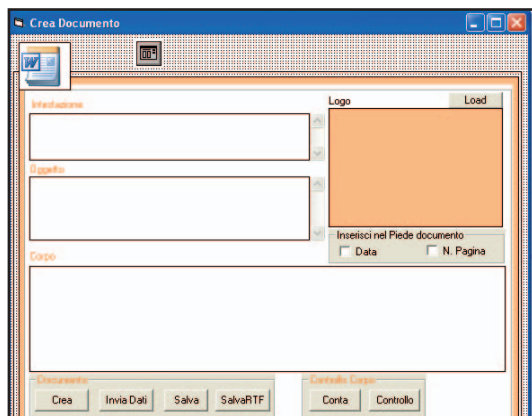



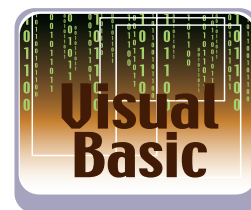
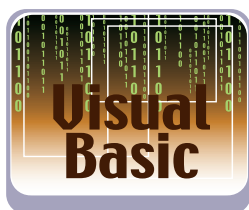
Fig. 3: ***La Form del generatore di documenti Word***

IL CODICE DEL MODELLO WORD

2 Create un modello Word, nominatelo *Modello.dot*, e salvatelo nella stessa directory che conterrà il progetto Visual Basic. In questo modello, prima del salvataggio, dovete inserire (nei moduli Visual Basic Word) le Macro: *WIntestazione*, che inserisce un'immagine ed una stringa nell'intestazione del documento, questi sono ricevuti con due parametri; *WPiede*, che in base ai valori di due parametri Boolean può impostare la data e il numero di pagina nel piede del documento infine la Macro *WCorpo*, senza parametri, ma che al suo interno definisce due Fields di documento. Come accennato i parametri e i Fields sono impostati da Visual Basic. Le Macro sono le seguenti.

```
Sub Wintestazione(para As String, immagine As String)
 ActiveWindow.View.Type = wdPrintView
 ActiveWindow.ActivePane.View.SeekView = _
                              wdSeekCurrentPageHeader
 If immagine <> "" Then
  Selection.InlineShapes.AddPicture FileName:= _
                           immagine, LinkToFile _
                  :=False, SaveWithDocument:=True
 End If
 Selection.EndKey Unit:=wdLine
 Selection.TypeText Text:=para
 ActiveWindow.ActivePane.View.SeekView = _
                                 wdSeekMainDocument
End Sub
```

Nella *WIntestazione*, innanzitutto, s'imposta il documento in Layout di stampa *(wdPrintView)*, altrimenti non si può selezionare l'intestazione e il Piè, dopo si abilita l'intestazione (*wdSeekCurrentPageHeader*) del documento e s'inseriscono l'immagine ed il testo passati come parametri. Notate che l'immagine è passata attraverso il suo Path e che con *SaveWithDocument:=True* si stabilisce che l'immagine verrà salvata nel documento.

```
Sub WPiede(data As Boolean, npagina As Boolean)
 ActiveWindow.ActivePane.View.Type = wdPrintView
 ActiveWindow.ActivePane.View.SeekView =
                              wdSeekCurrentPageFooter
 If data Then
  Selection.Fields.Add Range:=Selection.Range,
                                  Type:=wdFieldDate
 End If
 If npagina Then
  With Selection.Sections(1) .Headers(1).PageNumbers
   .NumberStyle = wdPageNumberStyleArabic
   .HeadingLevelForChapter = 0
   .IncludeChapterNumber = False
   .ChapterPageSeparator = wdSeparatorHyphen
   .RestartNumberingAtSection = False
   .StartingNumber = 0
  End With
  Selection.Sections(1).Footers(1).PageNumbers.Add
                           PageNumberAlignment:= _
          wdAlignPageNumberRight, FirstPage:=True
 End If
 If ActiveWindow.View.SplitSpecial <>
                                    wdPaneNone Then
  ActiveWindow.Panes(2).Close
 End If
```

```
  ActiveWindow.ActivePane.View.SeekView =
                                wdSeekMainDocument
End Sub
Sub Wcorpo()
 Selection.TypeParagraph
 Selection.Fields.Add Range:=Selection.Range, _
 Type:=wdFieldEmpty, Text:="DOCVARIABLE
                                  ""VarBody1""", _
 PreserveFormatting:=True
 Selection.EndKey Unit:=wdLine
 Selection.TypeParagraph
 Selection.TypeParagraph
 Selection.Fields.Add Range:=Selection.Range, _
 Type:=wdFieldEmpty, Text:="DOCVARIABLE
                                  ""VarBody2""", _
 PreserveFormatting:=True
End Sub
```

In base a quello illustrato s'intuisce che la procedura *WCorpo* crea due Fields di Documento associati alle variabili nominate *VarBody1* e *VarBody2*. I due Fields sono divisi da due righe vuote *(Selection.TypeParagraph)*.

Fig. 4: ***Un documento Word dopo l'invio dei dati***

Notate che *Selection.EndKey Unit:=wdLine* equivale alla selezione del tasto *Fine* cioè sposta il cursore alla fine della riga. Questo è necessario per evitare che, con l'inserimento della nuova linea, il *Field VarBody1* venga portato sotto.

ACTIVEX E RACCOGLITORE OFFICE

I componenti ActiveX, che hanno soppiantato la tecnologia *OLE*, rispecchiano in modo più realistico le funzioni di Visual Basic per lo sviluppo semplice e rapido di componenti software affidabili che interagiscono tra loro. Gli Activex possono essere raggruppati in oggetti raccoglitori.
Il raccoglitore di Activex principali sono: *Microsoft Internet Explorer* e *Microsoft Office Binder* (raccoglitore di Office).
Quest'ultimo può essere installato tramite il CD di Microsoft Office.
Il raccoglitore di Office da programma si controlla con gli oggetti: *Binder,* che rappresenta un raccoglitore office; *Section* che rappresenta una parte di un singolo documento, cioè una sezione del raccoglitore e l'oggetto *Sections* che rappresenta l'insieme delle sezioni del raccoglitore.

IL CODICE DEL PROGETTO VB

3 Per punti presentiamo le parti principali del codice Visual Basic da inserire nella Form1.

a) Nella parte dichiarativa e nella *Form_Load* prevediamo il seguente codice.

```
Dim objWor As Object
'Oggetto Application
Dim objDoc As Object
'Oggetto Document
Dim DocumentName as String
'Il nome del Template
Private Sub Form_Load()
 TxtCorpo.AutoVerbMenu = True
End Sub
```

Le tre variabili precedenti sono *Object*, poiché nel progetto utilizziamo la *CreateObject*, e non referenziamo direttamente le librerie di Office (per questo motivo nelle procedure sono state definite alcune costanti di Word).

b) Dopo aver inserito i dati nella form per creare il documento bisogna cliccare il pulsante *Crea*. La *Crea_Click* è la seguente.

```
Private Sub Crea_Click()
 DocumentName = App.Path + "/Modello.dot"
 Set objWor = CreateObject("Word.Application")
 objWor.Documents.Add Template:= DocumentName, _
 NewTemplate:=False
 Set objDoc = objWor.ActiveDocument
 MsgBox "Il Documento Word Creato è" _
 " in secondo piano", vbInformation
 objWor.Visible = True
 objWor.WindowState = 1
End Sub
```

Nella procedura dopo la creazione dell'applicazione Word è inserito un nuovo documento, nella collezione dei documenti, poi l'oggetto *objDoc* si fa referenziare al documento attivo.

c) La procedura seguente, invece serve per avviare l'esecuzione delle Macro e per passare i valori ai Fields.

```
Private Sub Macro_Click()
 Const wdGoToField = 7
 On Error GoTo errore
 If objDoc Is Nothing Then
  Crea_Click
 End If
 objDoc.Application.Run
                    MacroName:="Wintestazione", _
 varg1:=Me.TxtIntestazione, varg2:=
```

```
                         CommonDialog1.FileName
 If (Me.Checkdata.Value Or Me.Checkpagine.Value)
                                           Then
  objDoc.Application.Run MacroName:="WPiede", _
  varg1:=Me.Checkdata.Value,
                         varg2:=Me.Checkpagine.Value
 End If
 objDoc.Application.Run MacroName:="Wcorpo"
 objDoc.Variables.Add Name:="VarBody1", _
 Value:="Oggetto: " + Me.Txtoggetto
 objDoc.GoTo What:=wdGoToField, _
 Which:=wdGoToNext, Count:=1,
                              Name:="DOCVARIABLE"
 objDoc.Variables.Add Name:="VarBody2", _
 Value:=Me.TxtCorpo.Text
 objDoc.Fields.Update
 objWor.Visible = True
 objWor.WindowState = 1
 objWor.Activate
Exit Sub
errore:
 MsgBox Err.Description
 Err.Clear
End Sub
```

Nella procedura, quindi, sono fatte le seguenti azioni: si eseguono le tre macro con *Run*, si associano le variabili di documento ai Fields, si fa l'Update.

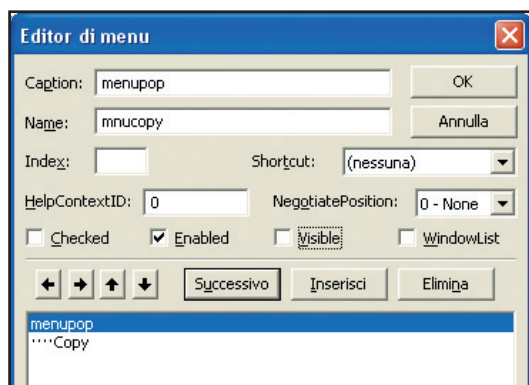



Fig. 5: ***Il menu a tendina del PopUp***

d) Le procedure per contare il numero di caratteri del campo TxtCorpo e per controllarne la sintassi sono le seguenti.

```
Private Sub Conta_Click()
 Const wdDialogToolsWordCount = 228
 Const wdDoNotSaveChanges = 0
 Dim Word As Object
 Set Word = CreateObject("Word.Application")
 Set doc = Word.Documents.Add
 Word.Selection.Text = TxtCorpo.Text
 Word.Dialogs(wdDialogToolsWordCount).Show
 Word.Visible = False
 doc.Close wdDoNotSaveChanges
```

```
 Word.Quit
End Sub
Private Sub controlla_Click()
 Const wdDialogToolsSpellingAndGrammar = 828
 Const wdDoNotSaveChanges = 0
 Dim Word, doc As Object
 Set Word = CreateObject("Word.Application")
 Set doc = Word.Documents.Add
 Word.Selection.Text = Me.TxtCorpo.Text
 Const wdItalian = 1040
 Word.Selection.LanguageID = wdItalian
 Word.Dialogs(wdDialogToolsSpellingAndGrammar
                                          ).Show
 Word.Visible = False
 If Len(Word.Selection.Text) <> 1 Then
  Me.TxtCorpo = Word.Selection.Text
 End If
 doc.Close wdDoNotSaveChanges
 Word.Quit
End Sub
```

e) Per caricare l'immagine nella PictureBox, per salvare il documento e il campo TxtCorpo in formato RTF, usiamo le seguenti procedure:

```
 Private Sub SalvaRTF_Click()
  Me.CommonDialog1.FileName = ""
  Me.CommonDialog1.Filter = "(*.rtf)|*.rtf|filertf"
  Me.CommonDialog1.ShowSave
  Me.TxtCorpo.SaveFile
                    Me.CommonDialog1.FileName, rtfRTF
  Exit Sub
 errore:
 MsgBox Err.Description
 Err.Clear
 End Sub
 Private Sub salvaword_Click()
 If objDoc Is Nothing Then
  MsgBox "Creare prima il documento Word"
  Exit Sub
 End If
  objWor.Dialogs(wdDialogFileSaveAs).Show
  objWor.Quit
 End Sub
 Private Sub load_Click()
  CommonDialog1.FileName = ""
  OpenFile
 End Sub
 Sub OpenFile()
 On Error GoTo errore
  If CommonDialog1.FileName = "" Then
   With CommonDialog1
    .ShowOpen
   End With
  End If
  If CommonDialog1.FileName <> "" Then
   Picturelogo.Picture = LoadPicture(
                              CommonDialog1.FileName)
```

FIELD

In un documento Word, per inserire, un Field di nome "DocVariable" si può utilizzare la finestra Field ... (in italiano Campo ...) selezionabile attraverso il menu Insert/Field (Inserisci/Campo). In questa finestra bisogna specificare il nome del Field (cioè "DocVariable") ed il nuovo nome che rappresenta il nome della variabile associata. Se su questa finestra si vuole vedere il codice di campo che sarà inserito nel documento, basta cliccare il pulsante Field Code.

```
 End If
 Exit Sub
errore:
MsgBox Err.Description
Err.Clear
End Sub
```

f) Dato che nel controllo RTF è possibile inserire anche delle immagini, il metodo più semplice per farlo è quello di usare i comandi detti *VerbMenu* cioè il copia ed incolla. In particolare copiamo l'immagine della PictureBox e la incolliamo sul RichTextBox. Per abilitare questi comandi sul RichTextBox usiamo *AutoVerbMenu = True* sulla PictureBox invece utilizziamo un *PopupMenu*. Il codice per far funzionare il *Popupmenu* è il seguente.

```
Private Sub mnucopy_Click()
 Clipboard.Clear
 Clipboard.SetData Me.Picturelogo.Picture
End Sub
Private Sub Picturelogo_MouseDown(Button As Integer, _
                Shift As Integer, X As Single, Y As Single)
 If Button = vbRightButton Then
  PopupMenu mnucopy
 End If
End Sub
```

Fig. 6: **La form in fase di esecuzione**

Ricordiamo che per il PopUpMenu è necessario un menu a tendina con almeno una voce di menu con un sottomenu, come mostrato in **Figura 6**.

USARE LE FUNZIONI DI EXCEL

Nel progetto inserito nel CD allegato alla rivista oltre agli esempi introdotti nel corso dell'articolo, trovate un esempio dedicato ad Excel, cioè la Form nominata *FrmExcel*. Essa contiene gli elementi - una *Listview*, un *CommonDialog*, un TextBox e tre pulsanti - che permettono di aprire il foglio Excel nominato *esempio_Excel.xls*, leggerne il contenuto e visualizzarlo su una ListView. Inoltre è presente un pulsante che permette di calcolare la somma di alcuni valori presenti nel foglio Excel, invocando la funzione Excel *Somma*. Di seguito riportiamo il codice contenuto in questo pulsante.

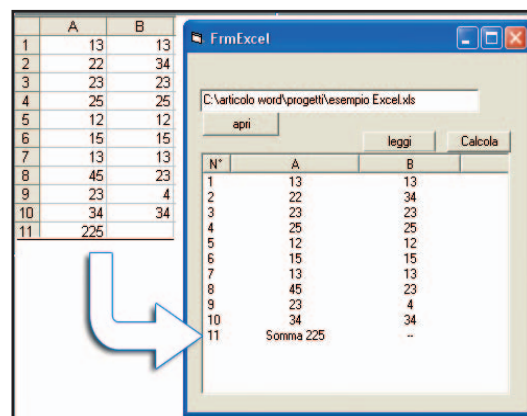



Fig. 7: **La form che utilizza le funzioni di Excel**

```
Private Sub calcola_Click()
 letterecolonneEx = Array("A", "B")
 FoglioExcel.Worksheets(1). _
 Range(letterecolonneEx(1) + "1:" + _
 letterecolonneEx(1) + CStr(righe + 1)).Select
 FoglioExcel.Worksheets(1). _
 Range(letterecolonneEx(1) + _
 CStr(righe + 1)).FormulaR1C1 = "
                                    =SUM(R[-10]C:R[-1]C)"
 Aggiungi righe + 1, "Somma " + _
 CStr(FoglioExcel.Worksheets(1).Range(
                                     letterecolonneEx(1) _
 + CStr(righe + 1))), "--"
 Set AppExcel = Nothing
 Set FoglioExcel = Nothing
Exit Sub
errore:
 MsgBox "Errore " & Err.Number & _
 vbCrLf & Err.Description

End Sub
```

Il resto del codice lo trovate nel CD.

CONCLUSIONI

L'articolo è un valido ausilio per chi vuole approfondire le conoscenze sulle applicazioni della Suite Office. Gli esempi sono relativi a Office XP ma possono essere utilizzati anche con Office 2003.

Massimo Autiero

WEB SERVICES CON VISUAL BASIC.NET

OGGI È POSSIBILE UTILIZZARE I WEB SERVICES PER SVILUPPARE APPLICAZIONI DISTRIBUITE IN MANIERA SEMPLICE, SICURA E VELOCE. VB.NET CONTIENE STRUMENTI EFFICACI PER QUESTA TECNOLOGIA, VEDIAMO QUALI

In questo articolo cominceremo a lavorare con una tecnologia relativamente nuova, quale è quella dei web service. Solitamente le realtà informatiche che si avvalgono di questo tipo di soluzioni sono aziende medio-grandi che a loro volta forniscono prodotti per aziende medio grandi. I motivi sono solitamente imputabili a due ragioni principali: la prima è che l'argomento web service viene solitamente presentato nei suoi aspetti di più basso livello senza invece evidenziarne le sue potenzialità ad alto livello. È normale quindi che in molte situazioni si preferisca non abbandonare le vecchie soluzioni, sulle quali si possiede un know-how già consolidato, piuttosto che intraprendere soluzioni apparentemente ostiche ed i benefici delle quali non appaiono così evidenti. La seconda va invece attribuita al fatto che i web service danno "il meglio di se stessi" in un'architettura distribuita, ed in questo caso si intende in particolar modo un'architettura geograficamente distribuita, cioè in una situazione peculiare di chi ha ad esempio più filiali dislocate su un ampio territorio che in qualche modo devono comunicare tra loro.

DALLA PROGRAMMAZIONE OBJECT ORIENTED A QUELLA SERVICE ORIENTED

La breve introduzione sopra ci servirà proprio per tentare di superare quei due punti evidenziati fornendo un punto di vista molto più operativo di quanto solitamente viene proposto. Partiamo da un'analogia con la quale, bene o male, tutti abbiamo fatto i conti. Una delle grandi rivoluzioni nel campo della programmazione è stata sicuramente l'introduzione degli oggetti, l'altra è quella della nascita e diffusione di internet. Ora provate a mettere insieme le due cose e che otterrete? Neanche a dirlo i web service! Uno dei concetti principali in ambito *OO* è che un oggetto può essere visto come

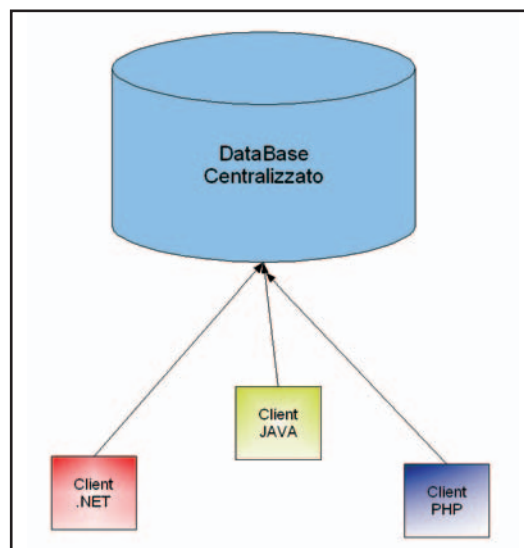



Fig.1: Un repository centralizzato con più "client" sparsi di cui non possiamo predire né il numero né la piattaforma.

una black box di cui è necessario conoscere solamente l'interfaccia senza doversi preoccupare della sua implementazione interna. Questo infatti ci permette di sfruttare un'infinità di librerie esterne (note anche come API) senza avere nessun tipo di informazione su come esse siano state implementate. D'altro canto è anche ovvio che non è possibile utilizzare delle API scritte in un linguaggio su un altro; ad esempio una libreria scritta in .NET non è utilizzabile su una piattaforma Java.
Ora poniamoci un caso d'uso concreto: supponiamo di aver necessità di dover sviluppare un'applicazione che interagisca con un database centralizzato dislocato in una qualsiasi punto della terra. Supponiamo inoltre che l'applicativo dovrà girare su piattaforme diverse (Windows, Linux etc...) e che implicitamente od esplicitamente venga imposto anche l'ambiente di sviluppo (.NET, Java, PHP, Perl). La **Figura 1** schematizza quanto detto fin qui. Una soluzione tra le più immediate da pensare ma meno efficiente dal punto di vista progettuale è quella di scrivere ogni volta l'applicazione ex novo, facendosi carico quindi sia della busi-

ness logic sia dell'interazione con il DBMS centrale. Se da una parte la business logic può essere in gran parte progettata astraendo dallo specifico linguaggio di programmazione, maggiori problemi vengono posti per quel che riguarda il secondo aspetto. Nel migliore dei casi avremmo bisogno degli opportuni driver per ogni specifico linguaggio che usiamo e dovremmo lasciare aperta un porta dedicata sia sul server centralizzato che ospita il DB sia sulla macchina che ospiterà l'applicazione con tutti i problemi di sicurezza che ne conseguono. Inoltre, pur ammesso di essere disposti a sobbarcarci questi inconvenienti, dobbiamo anche tener presente un aspetto economico: alcuni RDBMS commerciali richiedono (sia per motivi di efficienza che economici) che le interrogazioni a DB passino attraverso un loro client proprietario e, solitamente, qui entrano in gioco le cosiddette licenze call. Non sarebbe forse molto meglio creare "un'applicazione" che si occupi di esporre un'unica interfaccia che permetta l'interazione col DB e che allo stesso tempo sia usufruibile da qualsiasi linguaggio di programmazione? Beh, se questo è ciò che stiamo cercando, i web services sono la nostra soluzione.
Da questo esempio, che verrà sviluppato e implementato nel seguito dell'articolo, possiamo quindi fissarci fin d'ora una idea cardine: possiamo creare un'istanza di un qualsiasi web service come se istanziassimo una qualsiasi classe da una libreria esterna e utilizzarla a nostro web service.

BREVE OVERVIEW SUI WEB SERVICES

Abbiamo già detto che i web services permettono il dialogo tra diverse piattaforme e rendono possibile l'esportazione di componenti già esistenti verso client diversi. Il bello è che tutto ciò si basa su protocolli ben noti e di largo utilizzo.

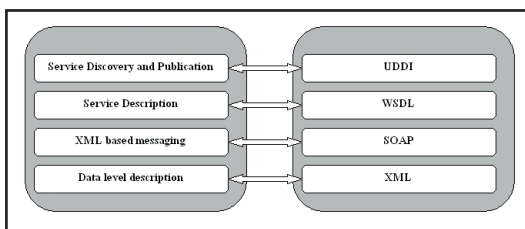



Fig.2: *Stack di protocolli su cui si basano i web service*

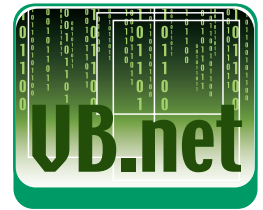
Come mostrato in **Figura 2** i web services si basano su quattro protocolli:

- **xml** è il metalinguaggio per antonomasia e viene usato per l'esportazione dei dati
- **soap** è lo standard che definisce il formato dei messaggi che saranno scambiati tra client e servizio
- **wdsl** è il protocollo utilizzato per la definizione dei parametri e dei metodi
- **uddi** è lo standard promosso dal consorzio *"uddi"* per facilitare la ricerca dei servizi in rete, su di esso si basano delle specie di motori di ricerca per trovare i web service.

Approfondire ogni singolo aspetto di questo stack non rientra certo tra gli obiettivi di questo articolo e tra l'altro, come già accennato, è uno dei motivi che rende poco gradevole tale argomento. A noi basta capire i principi che permettono l'astrazione e la comunicazione tra piattaforme non omogenee. Un web service in fondo non deve far altro che comunicare al client la propria interfaccia in modo tale che quest'ultimo possa al momento opportuno invocare i metodi che desidera. Ciò è reso possibile grazie al WSDL (*Web Services Description Language*); per rimanere nel concreto vediamo fin da ora come è strutturato il messaggio WSDL del web service messo a disposizione da google.
In particolare, come ovvio, Google mette a disposizione un web service che consente di effettuare ricerche come se si interrogasse dal proprio browser. Per ragioni di brevità analizzeremo solo alcuni tratti del file XML che descrive il messaggio WSDL, ad ogni modo chi volesse approfondire quest'aspetto può scaricarsi le API del web service di google dal sito *http://www.google.com/apis/download.htm*. La prima informazione che si ottiene è la definizione:

```
<definitions name="GoogleSearch"
        targetNamespace="urn:GoogleSearch"
        xmlns:typens="urn:GoogleSearch"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/
  " xmlns:soapenc="http://schemas.xmlsoap.org/soap/
  encoding/" xmlns:wsdl="http://schemas.xmlsoap.org
  /wsdl/" xmlns="http://schemas.xmlsoap.org/wsdl/">
```

Poi si passa a comunicare le classi che il web service mette a disposizione, ad esempio una classe che Google mette a disposizione è quella dei risultati provenienti dall'interrogazione:

```
<xsd:complexType name="GoogleSearchResult">
  <xsd:all>
    <xsd:element name="documentFiltering"
                        type="xsd:boolean"/>
    <xsd:element name="searchComments"
                        type="xsd:string"/>
    <xsd:element name=
      "estimatedTotalResultsCount" type="xsd:int"/>
    <xsd:element name="estimateIsExact" type=
                        "xsd:boolean"/>
    <xsd:element name="resultElements"
```

WINDOWS O LINUX?

Chi vi scrive è tendenzialmente abituato a sviluppare su piattaforme Linux, visto però che l'intento di questo articolo è quello di far sperimentare i web service al maggior numero di lettori possibile si è optato per la piattaforma più diffusa. Ci si può comunque avvalere di strumenti diversi ed ottenere i medesimi risultati. In particolare una tra le tante scelte che mi sento di consigliare è una architettura *LATJE*, ossia Linux Apache Tomcat Java Eclipse. Per essere un po' più precisi Apache sarà il vostro web server, Tomcat il container, Java il linguaggio di programmazione ed Eclipse l'IDE per sviluppare in tutta comodità. Inoltre, per sfruttare al massimo le possibilità di Eclipse suggerisco di installare anche il plugin *Lavadora* all'indirizzo http://lavadora.sourceforge.net/.

```
type="typens:ResultElementArray"/>
<xsd:element name="searchQuery"
type="xsd:string"/>
<xsd:element name="startIndex" type="xsd:int"/>
<xsd:element name="endIndex" type="xsd:int"/>
<xsd:element name="searchTips"
type="xsd:string"/>
<xsd:element name="directoryCategories"
type="typens:DirectoryCategoryArray"/>
<xsd:element name="searchTime"
type="xsd:double"/>
</xsd:all>
</xsd:complexType>
<xsd:complexType name="ResultElement">
<xsd:all>
<xsd:element name="summary"
type="xsd:string"/>
<xsd:element name="URL" type="xsd:string"/>
<xsd:element name="snippet" type="xsd:string"/>
<xsd:element name="title" type="xsd:string"/>
<xsd:element name="cachedSize"
type="xsd:string"/>
<xsd:element name="relatedInformationPresent"
type="xsd:boolean"/>
<xsd:element name="hostName"
type="xsd:string"/>
<xsd:element name="directoryCategory"
type="typens:DirectoryCategory"/>
<xsd:element name="directoryTitle"
type="xsd:string"/>
</xsd:all>
</xsd:complexType>
```

Come si può notare *GoogleSearchResult* oltre che contenere campi primitivi contiene a sua volta altre classi (ossia *complexType*) che a loro volta vengono definiti in maniera analoga. Alla fine di questa catena dovremmo comunque avere tutti dati serializzabili (proprio perché dovranno essere fatti viaggiare su internet) e nella grande maggioranza dei casi essi sono di tipo primitivo. Anche per quel che riguarda i metodi abbiamo una notazione del tutto simile:

```
<message name="doGoogleSearch">
<part name="key"          type="xsd:string"/>
<part name="q"            type="xsd:string"/>
<part name="start"        type="xsd:int"/>
<part name="maxResults"   type="xsd:int"/>
<part name="filter"       type="xsd:boolean"/>
<part name="restrict"     type="xsd:string"/>
<part name="safeSearch" type="xsd:boolean"/>
<part name="lr"           type="xsd:string"/>
<part name="ie"           type="xsd:string"/>
<part name="oe"           type="xsd:string"/>
</message>
<message name="doGoogleSearchResponse">
<part name="return"
type="typens:GoogleSearchResult"/>
</message>
```

Ecco ad esempio come viene definito il metodo che permette di effettuare una ricerca e riceverne i risultati.

SVILUPPARE UN WEB SERVICE

Ora che abbiamo fatto un breve excursus sui web service vediamo come sia possibile ed estremamente semplice svilupparne uno. Prima di fare ciò facciamo luce sugli strumenti necessari per sviluppare e “far girare” un web service.
Questo articolo è stato pensato per essere implementato su una piattaforma Windows che fosse la più accessibile e diffusa possibile. Ciò di cui avremo bisogno sarà un web server (da non confondere con il web service che andremo noi ad implementare!) come IIS; esso è già presente di default sulla maggior parte delle installazioni di windows XP Professional, la piattaforma .NET, e possibilmente anche Visual Studio che ci semplificherà notevolmente la fase di sviluppo. Inoltre, per rendere immediatamente possibile effettuare i nostri test in locale, utilizzeremo il Database NorthWind messo a disposizione da Access. Tornando al nostro progetto iniziale si tratterà quindi di inserire un layer suppletivo tra il database e le applicazioni client che puntano ad esso.
Tale layer suppletivo sarà, come mostrato in

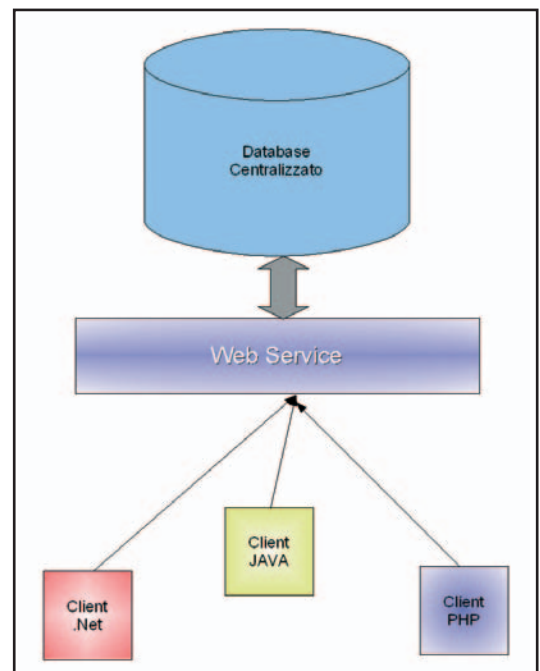



Fig.3: *Architettura con un web service che si frappone tra il database ed i vari client*

Figura 3, proprio rappresentato dal nostro web service che andremo adesso ad implementare.

IMPLEMENTAZIONE DEL WEB SERVICE

Possiamo sicuramente considerare un web service come una classe qualsiasi messa a disposizione da .NET. In particolare per creare un web service basterà ereditare da *System.Web.Services .Web-Service*. Quindi nel nostro caso avremmo:

```
Public Class Service1
    Inherits System.Web.Services.WebService
```

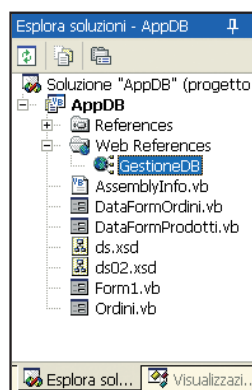



***Fig.4:** solution explorer dopo aver aggiunto la web reference*

Quando creiamo un web service con Visual Studio verrà creata una directory sotto quella principale di IIS, che per intenderci di default si tratta di *C:\ inetpub\wwwroot.* A questo punto possiamo implementare il nostro web service come se stessimo programmando una qualsiasi classe, con un solo piccolo accorgimento ulteriore.

Solitamente quando ci si occupa di programmazione Object Oriented in fase di progettazione si decide quali metodi esporre come pubblici, protetti o privati. Se ci si occupa di web service è necessario specificare anche quali metodi siano visibili "lato web", ossia quali metodi siano utilizzabili, o meglio invocabili, dai client che si collegheranno. Sulla piattaforma .NET questo viene ottenuto utilizzando la label (o etichetta) WebMethod. Prima di vedere più nel dettaglio il codice facciamo un'ultima considerazione. Abbiamo detto che il nostro web service funge da layer intermezzo tra database ed applicazioni client. Per ottenere questo risultato possiamo progettarlo seguendo due filosofie progettuali diverse. La prima è quella di realizzare un web service "su misura" del database; ad esempio se prendiamo in considerazione il database NorthWind e vogliamo rendere disponibili alcune informazioni sui clienti presenti nella tabella Clienti dovremmo ricevere in ingresso al metodo il nome, implementare all'interno la logica di interrogazioni e restituire i risultati. Se da un lato questa soluzione è molto sicura perché il client non si occuperà mai del database, dall'altra presuppone che ogni database abbia il suo web service specifico, perché cambiando il DB dovrà cambiare tutta la logica che sta dietro il web service stesso. Naturalmente chi vuole offrire un prodotto "già confezionato" tenderà solitamente a seguire una strada di questo tipo. Se però siamo comunque sempre noi anche a fornire le applicazioni client ci converrà tenere la soluzione web service il più aperta possibile e spostare la logica dal lato dell'applicazione. Come avrete sicuramente intuito sarà questa seconda soluzione che andremo a sviluppare. Per tenere quindi il nostro web service il più generale possibile, faremo in modo che i suoi metodi si occupino di ricevere in ingresso un generico comando SQL, ad esempio un web method utile potrebbe essere:

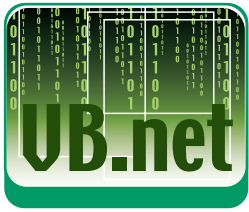
***Fig.5:** Finestra principale dell'applicazione client.*

```
<WebMethod()> _
  Public Function query(ByVal strSql As String,
                 ByVal strConnect As String) As DataSet
     Dim oDataSet As New DataSet
     Dim objConn As New
```

MA COME SI TROVANO I WEB SERVICE?

UDDI *(Universal Description, Discovery and Integration)* è un protocollo, nato da un consorzio formato tra gli altri da Microsoft, IBM e Ariba, che permette di localizzare i web services di proprio interesse. In breve UDDI non è altro che un protocollo comune per la ricerca di Web Services, che ci può aiutare a trovare soluzioni in breve tempo e facendolo da un'unica e sola fonte. Più precisamente UDDI si suddivide in 3 differenti elenchi:

- **Pagine Bianche: contengono le informazioni anagrafiche delle imprese registrate**
- **Pagine Gialle: attraverso l'utilizzo di opportune tassonomie (es: *UN/SPSC*) vengono descritti i prodotti ed i servizi offerti**
- **Pagine Verdi: esplicitano i termini tecnici attraverso i quali basare lo scambio informativo diretto tra i partner.**

A questo riguardo consiglio due siti che ogni sviluppatore e/o utilizzatore di web service non può esimersi dal consultare: il primo è http://www.uddi.org/, che è il sito ufficiale del consorzio ed il sito http://www.xmethods.org/ che rappresenta un ottimo motore di ricerca per chi voglia trovare i web service adatti alle proprie applicazioni.

```
                              OleDb.OleDbConnection(strConnect)
      Try
         objConn.Open()
         Dim objcommand As New
                        OleDb.OleDbCommand(strSql, objConn)
         Dim oAdapter As New
                        OleDb.OleDbDataAdapter(objcommand)
                                oAdapter.TableMappings.Add(
                                          "Table", "Table0")
         oAdapter.Fill(oDataSet)
      Catch ex As Exception
         Throw New Exception(ex.Message, ex)
      Finally
         objConn.Close()
      End Try
      Return oDataSet
  End Function
```

Come si può notare il web method in questione riceve una stringa di connessione al database ed il comando SQL da eseguire. Grazie ad ADO possiamo poi effettuare la query e restituire il dataset. Possiamo anche prevedere un metodo che esegua un *INSERT* od un *UPDATE* invece che una query, in questo caso abbiamo:

```
  <WebMethod()> _
    Public Function nonQuery(ByVal command As
               String, ByVal strConnect As String) As Integer
      Dim objConn As New
                        OleDb.OleDbConnection(strConnect)
       Try
         objConn.Open()
         Dim objcommand As New
                   OleDb.OleDbCommand(command, objConn)
         Return objcommand.ExecuteNonQuery
      Catch ex As Exception
         Throw New Exception(ex.Message, ex)
      Finally
         ' chiusa connessione
         objConn.Close()
       End Try
 End Function
```

Naturalmente il parametro restituito questa volta sarà il numero di record coinvolti nel comando.

DAL LATO DEL CLIENT

Come abbiamo visto il nostro web service con poche righe di codice è già pronto per essere usato da una qualsiasi applicazione client, cosa che ci accingeremo immediatamente a fare. Dopo aver aver iniziato un nuovo progetto di tipo Windows Application dovremo aggiungere una web reference proprio come se si trattasse di una libreria esterna. L'indirizzo che ci verrà chiesto nel nostro caso sarà: *http://localhost /Mercurio-WS_GestioneDB/Service1.asmx*.
A questo punto dobbiamo avere una situazione del progetto simile a quella riportata in **Figura 4**. Il nostro client prevede una finestra principale che permette di effettuare operazioni di ricerca e di visualizzare i dati ricevuti dal web service. Quest'ultima operazione può essere compiuta grazie al seguente codice:

```
 Private Sub query(ByVal command As String)
       Dim ws As New GestioneDB.Service1
       Dim ds As System.Data.DataSet =
                      ws.query(command, Me.connect)
       DataGrid1.SetDataBinding(ds, "Table0")
       DataGrid1.Refresh()
    End Sub
```

Come potete notare non si è fatto altro che istanziare un oggetto del nostro web service per poi chiamare il web method opportuno.

Fig.6: *Una form per gli ordini*

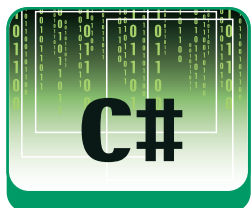
Il comando SQL viene generato da un'altra finestra come quella riportata in **Figura 6**, naturalmente potete inventarvi tutte le form che ritenete più opportune.
La creazione del comando sql può essere effettuato come segue:

```
sqlCommand As String = "SELECT * FROM Ordini
WHERE Ordini.IDCliente LIKE '"
.....
sqlCommand += TextBox1.Text & "%'"
```

CONCLUSIONI

Con questo articolo spero di aver suscitato il vostro interesse e la vostra curiosità nei riguardi di questo tipo di tecnologia e di essere riuscito a dimostrare come ormai i web service siamo alla portata di chiunque.

MICROSOFT MESSAGE QUEUE PROGRAMMING

ABBIAMO VISTO NEL NUMERO PRECEDENTE UNA OVERVIEW ARCHITETTURALE DEL PRODOTTO. IN QUESTO ARTICOLO VEDREMO MSMQ ALL'OPERA UTILIZZANDO LE VARIE OPZIONI DI DELIVERY

Nell'articolo precedente (MSMQ: Overview Architetturale) abbiamo cercato di chiarire con due esempi il ruolo di un sistema di messaggistica applicativa asincrona per poi vederne la forma più semplice di invio e ricezione di un messaggio tramite le classi .NET che sfruttano Microsoft Message Queue. In questo articolo proseguiamo con vari esempi applicativi che utilizzano le diverse opzioni che MSMQ mette a disposizione. Prima di iniziare è bene ricordare che l'applicazione sender non necessità di code definite in locale per appoggiare i messaggi quando non è connessa; l'applicazione sender scrive sempre direttamente in una coda remota senza preoccuparsi della connettività verso la coda. MSMQ tiene il messaggio in locale (in una coda di sistema) per poi spedirlo nella coda remota appena ritrovata la connettività: non occorre quindi definire una coda locale sul palmare dell'agente commerciale visto nell'articolo precedente; l'applicazione scriverà sempre nella coda aziendale e sarà MSMQ ad incaricarsi della spedizione non appena il palmare sarà connesso alla struttura aziendale. Partiamo subito con un esempio che andremo a commentare nelle varie sezioni:

```
…
using System.Messaging;
…
try
{
    using(MessageQueue mq = new MessageQueue(
                        "@gondor\private$\testmsmq"))
    {
        Message messaggio = new Message();
        messaggio.Body = "Ciao dal Sender";
        messaggio.Priority = MessagePriority.High;
        messaggio.Recoverable = true;
        mq.Send(messaggio);
    }
    MessageBox.Show("Inviato!");
}
catch (MessageQueueException ex)
{
    MessageBox.Show("Errore invio");
}
```

REQUISITI

Conoscenze richieste
Architettura di MSMQ. Si veda ioProgrammo N° 101

Software
.NET Framework 1.x o 2.0 per testare il codice presentato

Impegno

Tempo di realizzazione

PRIORITÀ DEI MESSAGGI

L'esempio di codice è identico a quanto mostrato nell'articolo precedente con alcune differenze che consentono di sfruttare le opzioni di delivery disponibili. Come abbiamo accennato è possibile dare una priorità ai messaggi: è disponibile l'enum *System.Messaging.MessagePriority* per impostare la priorità al messaggio; in ordine di "priorità" le opzioni disponibili sono *Highest, VeryHigh, High, AboveNormal, Normal, Low, VeryLow, Lowest*. È bene ribadire che la priorità non è assoluta: non è assolutamente detto che un messaggio a priorità bassa sia letto (ricevuto dal receiver) dopo un messaggio con priorità alta. Il motivo è semplice: se un'applicazione spedisce un messaggio a priorità bassa e subito dopo un messaggio a priorità alta e l'applicazione receiver è libera (e esiste connettività verso la coda) riceverà subito il messaggio a priorità bassa che sarà processato prima del messaggio successivo. Ricordatevi che questo non rappresenta un problema… anzi!

MESSAGGI RECOVERABLE

La seconda proprietà utilizzata nell'esempio è *Recoverable*. Un messaggio non recoverable (il default) viene tenuto in RAM dalla macchina che lo deve inviare verso la coda di destinazione e da tutte le macchine che agiscono da router verso la coda di destinazione (un server MSMQ può agire da MSMQ Router per instradare il messaggio in configurazioni di rete complesse). Questo significa che se una di queste macchine va in crash, oppure il servizio MSMQ viene riavviato oppure più semplicemente si fa un reboot della macchina, il messaggio (anzi i messaggi) "normali" vengono persi e quindi non spediti verso la coda di destinazione. L'impostazio-

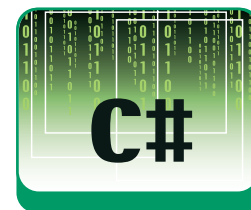
ne di default è utile per tutte le applicazioni che, ad esempio, devono avvertire un'applicazione receiver in tempi ragionevoli sullo stato applicativo. L'esempio più classico è in campo industriale: molto spesso le varie macchine di produzione inviano il loro stato a applicazioni che tramite gauge, indicatori di livello, grafici, mostrano queste informazioni ad un operatore a video. MSMQ potrebbe essere un ottimo sistema per far colloquiare il software della macchina di produzione con l'applicazione di monitoraggio. Se la connettività fra le macchine viene interrotta o, più semplicemente, viene riavviato il PC della macchina industriale è inutile tenere i messaggi informativi perché non avrebbero più significato passato il tempo necessario al reboot. In tutti gli ambiti in cui è importante che tutti i messaggi spediti vengano recapitati a destinazione (e sono sicuramente più frequenti gli scenari in cui questo comportamento è necessario rispetto all'esempio precedente) è possibile impostare la proprietà *Recoverable* a *true* per fare in modo che le macchine coinvolte nella spedizione del messaggio verso la coda di destinazione appoggino i messaggi stessi su disco. Com'è intuibile i messaggi Recoverable sopravvivono ai reboot, ai crash applicativi e ai restart dei servizi di Message Queue. È altrettanto ovvio che la spedizione sarà un minimo più lenta e consumi più risorse rispetto ai messaggi *Recoverable=false*. Ancora una volta MSMQ fornisce l'infrastruttura necessaria per gestire entrambe le problematiche senza dover scrivere codice complesso per raggiungere questi obiettivi.

TIMEOUT IMPOSTABILI

Complichiamo ancora un attimo l'esempio:

```
…
using System.Messaging;
…
try
{
    using(MessageQueue mq = new MessageQueue(
                            "@gondor\private$\testmsmq"))
    {
        Message messaggio = new Message();
        messaggio.Body = "Ciao dal Sender";
        messaggio.Priority = MessagePriority.High;
        messaggio.Recoverable = true;
        messaggio.TimeToReachQueue =
                            TimeSpan.FromMinutes(10);
        messaggio.TimeToBeReceive =
                            TimeSpan.FromMinutes(20);
        mq.Send(messaggio);
    }
    MessageBox.Show("Inviato!");
}
```

```
catch (MessageQueueException ex)
{
    MessageBox.Show("Errore invio");
}
```

Le ultime due proprietà impostate sull'oggetto messaggio nel codice precedente consentono invece di marcare il *messaggio* con due attributi utilissimi ancora una volta nell'esempio citato dell'automazione industriale: i messaggi di stato delle macchine che vengono inviati all'applicazione di monitoraggio hanno un senso se visualizzati entro un certo periodo di tempo. Con la proprietà *TimeToReachQueue* si indica il tempo, in secondi, che il messaggio può impiegare per raggiungere la coda di destinazione: in pratica il tempo valido entro cui il messaggio ha senso che venga spedito nella coda di destinazione. Proviamo a chiarire con un altro esempio: un'applicazione che monitorizza le quotazioni di borsa invia al database che poi alimenta il sito pubblico consultabile dai clienti aggiornando le quotazioni ogni 10 minuti. È assolutamente inutile inviare i messaggi con le quotazioni delle ore 13:00, 13:10, 13:20, 13:30 se la connettività fra l'applicazione di monitoraggio e il sito web si interrompe dalle 13:05 alle 13:54. Se ogni messaggio con la quotazione di un titolo ha la proprietà *TimeToReachQueue* impostata allo scadere dei dieci minuti successivi *(TimeSpan.FromMinutes(10))* MSMQ eviterà di spedire messaggi "inutili" al sito web. La seconda proprietà *(TimeToBeReceived)*, consente invece di impostare la validità temporale del messaggio stesso; il significato è il seguente: è inutile che l'applicazione Receeiver legga un messaggio se sono passati più di X minuti, secondi, ore etc da quando è stato creato. In poche parole l'applicazione Sender indica entro quando il messaggio deve essere letto dall'applicazione Receiver. È inutile dire che questa impostazione viene utilizzata più frequentemente rispetto alla precedente in quanto molto spesso si pensa a qual è il tempo massimo di vita di un messaggio. Per torna-

MICROSOFT MESSAGE QUEUE

Per chi avesse perso la puntata precedente è bene ricordare che MSMQ è il servizio di Microsoft che consente di inviare messaggi da una macchina ad un`altra tramite un sistema di code asincrono. In sostanza è possibile sviluppare un software che utilizza un sistema di comunicazione simile a quello delle segreterie telefoniche. L'esempio più classico di utilizzo è quello dell'ecommerce. Un cliente effettua un pagamento su un sito web con carta di credito, la connessione con la banca è assente, la soluzione classica sarebbe quella di mostrare un messaggio d'errore all'utente, la soluzione con MSMQ prevede che la transazione venga mantenuta in una coda e che essa venga iniziata non appena la connessione con la banca viene ripristinata, inviando poi in modo asincrono una mail al cliente avvertendolo che il pagamento è avvenuto correttamente.

re all'esempio specifico dell'automazione industriale citato precedentemente, oltre alla proprietà *Recoverable* impostata a false, si imposterà su ogni messaggio il tempo in cui il valore di stato della macchina di produzione ha un senso. La proprietà *TimeToReachQueue* sembra che non abbia senso: in realtà è utilissima per non appesantire il sistema con la trasmissione di messaggi che non hano senso. Molto spesso infatti i due timeout vengono impostati allo stesso valore. In termini funzionali un messaggio che non ha raggiunto la coda di destinazione entro il *TimeToReachQueue* viene cancellato automaticamente da MSMQ dalla sua coda di spedizione interna. Un messaggio che, anche se avesse raggiunto la coda di destinazione, non venga letto entro il tempo impostato con *TimeToBeReceived* viene anch'esso cancellato dalla coda di destinazione. Dovrebbe risultare ovvio che un messaggio scaduto come *TimeToBeReceived* non inizi la sua corsa verso la coda di destinazione e che il *TimeToBeReceived* non ha senso venga impostato a un valore inferiore rispetto a *TimeToReachQueue*.

JOURNALING E DEAD-LETTER

Se iniziamo a lavorare con i parametri di delivery appena accennati per uno scenario applicativo è probabile che lo scenario stesso ci imponga un controllo sui messaggi spediti, i messaggi positivamente recapitati (i messaggi arrivati e/o letti) e i messaggi non spediti (per *timeout*) o non letti (entro il *timeout*). Fortunatamente tutte queste opzioni sono contemplate dai servizi esposti da MSMQ. È importante sottolineare da subito, però, che tutte queste opzioni forniscono informazioni sui messaggi spediti, non spediti, non ricevuti, letti, ma non ci danno nessuna informazione sull'esito dell'operazione fatta dal receiver sul messaggio stesso: il passaggio successivo sarà infatti quello di ottenere una risposta da parte del *Receiver* sull'esito dell'operazione che lui stesso ha compiuto in base al contenuto del messaggio. Procediamo con ordine. Se vogliamo tenere traccia di tutti i messaggi spediti è possibile abilitare il *Journaling*. Ogni coda creata in MSMQ ha un giornale di bordo dove può essere tenuta una copia di tutti i messaggi recapitati nella coda stessa. È possibile decidere per ogni messaggio, tramite la proprietà *UseJournalQueue*, se duplicarne il contenuto nel giornale di bordo. Inoltre ogni macchina MSMQ di tipo server o di tipo *Indipendent client* ha una coda di sistema che si trova sotto *System Queues* denominata *Journal Messages* dove tenere una copia di tutti i messaggi transitati dalla macchina. Si può abilitare il *journaling* per ogni messaggio tramite la proprietà indicata oppure definire il *journaling* a livello di coda: per ogni coda, dalla maschera delle proprietà. Inoltre è possibile limitare la dimensione del giornale di bordo in KB per evitare una crescita illimitata: è bene ricordare che se si usa il journaling a livello di coda ogni messaggio ricevuto viene anche copiato nel giornale di bordo quindi potenzialmente si può arrivare a diversi GigaBytes in poco tempo se i messaggi transitati sono molti e/o la loro dimensione è notevole. La seconda opzione per tracciare i messaggi è abilitare, tramite la proprietà *UseDeadLetterQueue* l'utilizzo di una coda di sistema dove recapitare tutti i messaggi impossibili da spedire. Ad esempio se per un messaggio scade il tempo impostato per raggiungere la coda di destinazione *(TimeToReachQueue)*, sulla macchina dell'applicazione *Sender* verrà copiato il messaggio nella coda di sistema identificabile con il nome di *Dead-Letter Messages*. Dall'interfaccia di amministrazione o da codice è possibile aprire tale coda per ispezionarne il contenuto e quindi sapere quali e/o quanti messaggi non sono stati recapitati.

ADMINISTRATION QUEUE

Volendo avere più controllo sull'esito dell'invio dei messaggi, oltre alle due tecniche appena citate *(Journaling* e *Dead-Letter)* si può impostare sul singolo messaggio la proprietà *AcknoledgeType* e la proprietà *AdministrationQueue*. La seconda proprietà indica una coda, definita dall'applicazione, dove recapitare l'esito di invio di ogni messaggio. In pratica, tramite l'abilitazione del journal avviene una copia del messaggio nel giornale di bordo, tramite l'utilizzo della dead-letter è possibile identificare i messaggi non recapitati, tramite l'impostazione della coda amministrativa *(AdministrationQueue)* è possibile indicare per ogni messaggio dove il motore di MSMQ debba recapitare una copia del messaggio stesso contenente anche l'esito dell'operazione. Vediamo il codice per chiarirci le idee:

```
…
using System.Messaging;
…
try
{
    using(MessageQueue mq = new MessageQueue(
                        @"gondor\private$\testmsmq"))
    {
        Message messaggio = new Message();
        messaggio.Body = "Ciao dal Sender";
        messaggio.Priority = MessagePriority.High;
        messaggio.Recoverable = true;
        messaggio.TimeToReachQueue =
                        TimeSpan.FromMinutes(10);
        messaggio.TimeToBeReceive =
                        TimeSpan.FromMinutes(20);
        messaggio.AdministrationQueue =  new
```

CODICE CHIARO

In tutti gli esempi di questi due primi articoli viene utilizzata la modalità più semplice di scrittura codice per rendere tutto più chiaro. Infatti non vengono usati componenti helper o dll esterne per centralizzare le righe di codice per la spedizione dei messaggi. Ovviamente per le applicazioni reali conviene sempre crearsi la propria libreria helper per centralizzare il codice che utilizza le classi del namespace System.Messaging esattamente come si fa per l'accesso ai dati.

```
                MessageQueue(@"nomemacchina\private$
                              \testmsmqAdminQueue");
        messaggio.AcknowledgeType =
                        AcknowledgeTypes.FullReceive;
        mq.Send(messaggio);
    }
    MessageBox.Show("Inviato!");
}
catch (MessageQueueException ex)
{
    MessageBox.Show("Errore invio");
}
```

Con il codice presentato è possibile indicare a MSMQ che dovrà recapitare un *messaggio* di acknoledgement per il nostro messaggio nella coda *"testmsmqAdminQueue"* (ovviamente la coda deve esistere). Nel nostro caso stiamo chiedendo un *FullReceive*, cioè vogliamo sapere se il messaggio è arrivato a destinazione ed è stato letto dall'applicazione receiver.
L'enum *System.Messaging.AcknowledgeTypes* consente di indicare:

1) *FullReachQueue*
2) *FullReceive*
3) *NegativeReceive*
4) *None*
5) *NotAcknowledgeReachQueue*
6) *NotAcknowledgeReceive*
7) *PositiveArrival*
8) *PositveReceive*

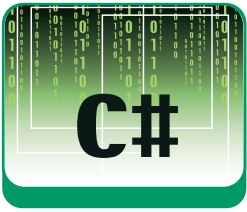
Senza dilungarsi nel significato di ognuna di queste impostazioni, facilmente identificabili nell'help sotto l'enum *System.Messaging.AcknowledgeTypes*, è importante capire che gli ultimi due indicano che vogliamo ricevere il messaggio, nella coda amministrativa indicata da codice, per sapere rispettivamente se il messaggio è arrivato nella coda di destinazione o è stato ricevuto dall'applicazione receiver. *NegativeReceive* indica che vogliamo sapere se il messaggio non è stato ricevuto, *FullReceive* indica che vogliamo sapere "tutto", vale a dire se il messaggio è stato ricevuto dall'applicazione *receiver* oppure se non è stato ricevuto dall'applicazione receiver. La coda amministrativa è una coda a tutti gli effetti, quindi si possono utilizzare tutti i metodi utilizzabili per qualunque coda: *Receive, GetAllMessages, GetMessageEnumerator, Exist, Create* e così via. Il messaggio recuperato con Receive (come un normale messaggio in una coda) espone la proprietà *Acknowledgement* con la quale è possibile capire cosa è successo al messaggio. Ancora una volta è disponibile l'enum *System.Messaging.Acknowled-gement* per capire meglio il tipo di ack ricevuto. *ReachQueue* indica che il messaggio ha raggiunto la coda di destinazione, *ReachQueueTimeout* indica invece che è scaduto il timeout indicato (*TimeToReachQueue*) per raggiungere la coda e quindi il messaggio non è stato recapitato. *Receive* indica che il messaggio ha avuto una receive positiva da parte dell'applicazione *Receiver,* mentre *ReceiveTimeout* indica che il messaggio non è stato ricevuto in quanto è scaduto il timeout impostato per la ricezione (*TimeToBeReceived*).

RESPONSE QUEUE

Come abbiamo accennato all'inizio dell'articolo, è importante sapere che tutte le opzioni citate fino ad adesso forniscono informazioni sui messaggi spediti, non spediti, non ricevuti, letti, ma non ci danno nessuna informazione sull'esito dell'operazione fatta dal receiver sul messaggio stesso. L'ultimo passaggio di questo articolo è ottenere una risposta da parte del Receiver sull'esito dell'operazione che lui stesso ha compiuto in base al contenuto del messaggio. Tramite questa risposta è possibile prendere provvedimenti come ad esempio avvertire l'utente che l'operazione che lui aveva (il passato è d'obbligo) tentato di fare (notare la parola "tentato") non è andata a buon fine.
Vediamo il codice dell'applicazione sender e dell'applicazione receiver per poi commentarli

```
...
using System.Messaging;
...
try
{
    using(MessageQueue mq = new
MessageQueue("@gondor\private$\testmsmq"))
    {
        Message messaggio = new Message();
        messaggio.Body = "Ciao dal Sender";
        messaggio.ResponseQueue = new
MessageQueue("@PEPPE\private$\testmsmqResponse");
        mq.Send(messaggio);
    }
    MessageBox.Show("Inviato!");
}
catch (MessageQueueException ex)
{
    MessageBox.Show("Errore invio");
}
```

Solitamente l'applicazione sender indica, tramite la proprietà *ResponseQueue* del messaggio dove vorrebbe che venissero recapitate le risposte sull'esito dell'operazione. È probabile che la coda di risposta stia sulla stessa macchina dell'applicazione sender: ad esempio, nello scenario dell'agente esterno che prende ordini con il palmare è probabile che l'applicazione sender sul palmare debba essere informata quando non è possibile creare un ordine per il clien-

te, quindi l'applicazione sender indicherà come *ResponseQueue* una coda locale sul palmare dove recapitare i messaggi. L'applicazione receiver analizza il messaggio in ingresso, prova a creare un ordine e in caso negativo utilizzerà la coda (sotto forma di proprietà del messaggio ricevuto) per inviare un messaggio che indica al sender "la richiesta di ordine xyz non è stata processata" e possibilmente i motivi del rifiuto (il cliente non ha pagato le ultime fatture…caso classico). In questo caso il codice del sender potrebbe essere rivisto in questo modo:

```
...
using System.Messaging;
...
try
{
    using(MessageQueue mq = new
        MessageQueue("@gondor\private$\testmsmq"))
    {
        Message messaggio = new Message();
        messaggio.Body = "Ciao dal Sender";
        messaggio.ResponseQueue = new
        MessageQueue("@NomeClient\private$\
                                testmsmqResponse");
        mq.Send(messaggio);
    }
    MessageBox.Show("Inviato!");
}
catch (MessageQueueException ex)
{
    MessageBox.Show("Errore invio");
}
```

Ovvero il codice sul palmare recupera dinamicamente il nome del device per formare la stringa che rappresenta la coda di risposta: in questo modo il codice è indipendente dal device su cui gira. L'applicazione Receiver invierà un messaggio di risposta sulla coda indicata dal client. Rimanendo sull'esempio semplice di invio di stringa nel body, ecco il codice dell'applicazione Receiver :

```
...
using System.Messaging;
...
try
{
    using(MessageQueue mq = new MessageQueue(
                "@gondor\private$\testmsmq"))
    {
        mq.Formatter = new XmlMessageFormatter(
                new Type[] { typeof(String) });
        // Ricezione del messaggio
        Message messageReceived = mq.Receive();
        String contenuto =
                (String)messageReceived.Body;
        // Processo il contenuto del messaggio...
        ....
        // Supponiamo non si possa fare l'operazione
        // Inviamo risposta al mittente
        try
        {
            using(MessageQueue mq = new
                        MessageQueue(
                messageReceived.ResponseQueue))
            {
                Message messaggio = new Message();
                messaggio.Body = "Risposta negativa al
                    messaggio " + messageReceived.Id;
                mq.Send(messaggio);
            }
            MessageBox.Show("Inviato!");
        }
        catch (MessageQueueException ex)
        {
            MessageBox.Show("Errore invio risposta");
        } }
}
catch (MessageQueueException ex)
{
    MessageBox.Show("Errore ricezione messaggio");
}
```

Nel codice precedente, l'applicazione Receiver, tenta di eseguire l'operazione e, in caso negativo, recupera dal messaggio la coda indicata (*messageReceived.ResponseQueue*) per la risposta; tramite gli stessi oggetti usati dal sender invia un messaggio nella coda indicata. Anche il receiver può utilizzare tutte le proprietà viste in questo articolo per sapere l'esito del messaggio di risposta: ad esempio può indicare un *AcknoledgeType*, un *TimeToReachQueue* e così via. La proprietà *Id* viene solitamente inviata nella risposta per indicare al client a quale messaggio si riferisce la risposta. Nello scenario che ci portiamo dietro dal primo articolo, il receiver potrebbe confezionare una risposta che riporta il numero di *"Richiesta Ordine"* rifiutata senza usare la proprietà *Id*. In altri scenari la coda di risposta è fissa e viene appoggiata su una macchina dove magari risiede un'applicazione preposta ad analizzare le risposte negative o positive dell'applicazione receiver e magari informa via mail chi di dovere. In questo caso il sender non compilerà la *ResponseQueue* che probabilmente verrà letta dall'applicazione Receiver da un file di configurazione locale.

CONCLUSIONI

MSMQ rappresenta uno strumento estremamente scalare per la cessione di code di messaggi e si pone come nodo centrale nella comunicazione asincrona.

Roberto Brunetti

Roberto Brunetti è un libero professionista del gruppo DevLeap (www.devleap.com) sul cui sito si trovano articoli e blog tecnici sulle tecnologie legate allo sviluppo software in .NET. E' specializzato in ASP.NET, Sviluppo mobile, Architetture distribuite e Visual Studio Team System. E' l'autore del libro ASP.NET Full Contact edito da Mondadori Informatica e numerose pubblicazioni su riviste del settore. Ha partecipato come speaker a numerose conferenze del settore.

BEANSHELL SCRIPTING: ESTENDIAMO JAVA

IN MOLTISSIMI SCENARI LA PRESENZA DI UN LINGUAGGIO NON COMPILATO PUÒ RAPPRESENTARE UNA RISORSA CHE CONSENTE ALL'UTENTE DI PERSONALIZZARE L'APPLICAZIONE SENZA DOVER COMPIERE SFORZI NOTEVOLI. UTILIZZIAMO QUESTA TECNICA...

Spesso esistono stringenti requisiti di rapidità nello sviluppo. Un aiuto può derivare dall'incorporare un motore di scripting all'interno di Java. Nell'articolo approfondiremo la soluzione proposta da BeanShell. Un linguaggio di scripting, quale PHP, JavaScript o lo stesso BeanShell presentato in questo articolo, si differenzia da un linguaggio compilato come Java per due principali ragioni: il modo in cui un programma viene eseguito e la tipizzazione delle variabili. Per quanto riguarda l'esecuzione, i linguaggi di scripting prevedono solitamente un'esecuzione tramite interprete. Questo componente software analizza il sorgente e ne esegue le varie istruzioni passo passo. I programmi scritti con linguaggi compilati devono invece essere trattati da un compilatore che produce un codice eseguibile direttamente da un processore fisico, come nel caso di C, o da una sorta di processore emulato via software, come avviene ad esempio con la macchina virtuale Java. Questa differenza è nella maggior parte dei casi di poca importanza per il programmatore, se non in relazione alla velocità d'esecuzione, ma la stessa cosa non si può certo dire della questione della tipizzazione delle variabili. In un linguaggio tipizzato è necessario dichiarare il tipo di ogni variabile utilizzata. Con queste informazioni il compilatore è in grado di controllare che le variabili siano utilizzate in modo consistente. Ad esempio se una funzione prevede un parametro di tipo intero ma in una chiamata è passato come parametro una variabile di tipo stringa, il compilatore rileva l'incongruenza, interrompere la compilazione ed avverte il programmatore. Nei linguaggi di scripting invece non sempre è necessario dichiarare una variabile prima di utilizzarla ed è possibile assegnarle valori di volta in volta di tipo differente: interi, stringhe, date, riferimenti ad oggetti, senza alcuna limitazione. Questa caratteristica permette di rinunciare a molto del formalismo imposto dai linguaggi compilati, permettendo sicuramente un avvio e una realizzazione più rapida di progetti di piccole dimensioni. Molti lettori avranno avuto sicuramente la possibilità di verificare quanto possa essere tedioso e lento sviluppare un servizio web di dimensioni limitate con Java e quanto sia invece immediato e semplice implementarlo con PHP. I linguaggi interpretati offrono inoltre tempi di deploy notevolmente inferiori. Un update di un servizio web realizzato con PHP si limita nella maggior parte dei casi ad un upload sul server della nuova versione del file sorgente ove sia stata apportata la correzione, mentre un update di un servizio web basato su Java richiede in linea di massima la compilazione, la creazione degli archivi e il deploy degli stessi sul server. La forza dei linguaggi di scripting nei piccoli progetti diventa però la loro debolezza nello sviluppo di sistemi più complessi. La compilazione e la tipizzazione difatti permettono di garantire la consistenza nelle assegnazioni delle variabili mentre con un linguaggio di scripting eventuali incongruenze si verificano solo a runtime. L'ideale sarebbe un linguaggio che permetta di essere utilizzato contemporaneamente come linguaggio compilato o come linguaggio di scripting a seconda delle esigenze. *BeanShell* mira a soddisfare questo bisogno fornendo la possibilità di programmare in Java con un linguaggio di scripting.

PREPARARE L'AMBIENTE DI LAVORO

Collegatevi al sito *http://www.beanshell.org/* e portatevi nella sezione "download". Da qui scaricate l'ultima versione di *BeanShell* che al momento della stesura dell'articolo è la *2.0b4*. Varie sono le distribuzioni offerte: una contiene tutti gli elementi di BeanShell in un unico *jar file*, mentre le altre offrono separatamente l'interprete e i vari componenti accessori. Per la realizzazione degli esempi proposti in questo articolo è consigliato l'utilizzo della libreria completa *bsh-2.0b4.jar*. Prima di avventurarci nell'utilizzo di BeanShell all'interno di un programma Java presentiamo qualche esempio. BeanShell viene fornito con un'interfaccia testuale all'interprete che permette di eseguire programmi stand alone. Fate partire l'interfaccia aprendo la console del sistema operativo e digitando il seguente comando,

avendo cura di sostituire <*path*> con il percorso per raggiungere il file *jar* di BeanShell appena scaricato.

```
java -jar <path>\bsh-2.0b4.jar
```

Si apre la finestra di BeanShell con un workspace attivo. Ciccate su *"File > capture System in/out/err"* per visualizzare nella finestra del workspace i messaggi prodotti dal programma che altrimenti sarebbero visibili solo nella console del sistema operativo. A questo punto ciccate su *"File > Workspace editor"*. Si aprirà un'altra finestra nella quale inseriremo i programmi BeanShell.

VARIABILI NON TIPIZZATE

In Beanshell non è necessario dichiarare preventivamente il tipo di una variabile, è sufficiente utilizzarla. Il sorgente BeanShell seguente da una prima idea della sintassi molto simile a quella di Java e mostra come possano essere usate variabili non tipizzate.

```
a = 5.5;
b = new Integer(5);
print(a+b);
a = "5.5";
b = "5";
print(a+b);
a = new int[]{1,2,3};
print(a);
```

Lanciamo il programma mediante la voce di menu *"Evaluate > eval in workspace"*. L'output prodotto è:

```
10.5
5.55
```

Come è possibile vedere dall'esempio precedente la variabile *"b"* è inizialmente di tipo Integer e poi di tipo *String*. Inoltre la prima somma viene calcolata tra un oggetto *Integer* ed un *int*. Nelle versioni di Java precedenti alla 1.5 sarebbe stato necessario convertire l'*Integer* in *int* mediante il metodo *intValue()*. BeanShell invece incorpora un meccanismo di autoboxing che converte automaticamente i tipi *wrapper* come *Boolean, Integer, Long* nei corrispondenti tipi primitivi sin dalle prime versioni. Nel secondo caso, essendo diventata *"b"* una stringa, la somma prende il significato di concatenazione.

UTILIZZO DI FUNZIONI

Negli script BeanShell è ovviamente possibile utilizzare funzioni. L'esempio seguente mostra una funzione che calcola la somma degli elementi contenuti in un array di valori interi.

```
sum(arr){
  sum = 0;
  for(x=0; x<arr.lenght; x++){
    sum = sum + arr[x];
  }
}
values = new int[]{3,43,23,-12};
print(sum(values));
```

Lanciamo il programma mediante la voce di menu *"Evaluate > eval in workspace"*. L'output prodotto è ovviamente 57.

INTEGRARE JAVA E BEANSHELL

Per vedere all'opera le potenzialità dell'integrazione tra Java e BeanShell sviluppiamo il core di quella che potrebbe essere un'applicazione reale. Una nuova catena di palestre richiede lo sviluppo di un sistema informativo per la gestione degli iscritti. In particolare il reparto marketing ha pensato di associare ad ogni cliente, per fidelizzarlo, un profilo tariffario che offra sconti sugli ingressi. Il committente per presentarsi in maniera aggressiva sul mercato prevede di proporre ai clienti profili tariffari sempre diversi ed è assolutamente necessario che tali profili siano resi disponibili in tempi brevissimi. Riuscire a soddisfare i bisogni di immediatezza del cliente potrebbe essere difficoltoso utilizzando esclusivamente Java poiché si dovrebbe magari implementare una nuova classe per ogni nuovo profilo tariffario pensato dal reparto marketing, farne il deploy, e così via. Potrebbe essere una buona idea realizzare la funzionalità di calcolo del costo di ingresso del biglietto con BeanShell, associando ogni profilo un programma. Non essendo richiesta nessuna ricompilazione si potrebbero così rendere gestibili da sistema molte tariffe senza cambiare minimamente il codice compilato.

STRUTTURA DELL'APPLICATIVO

La classe *Customer* rappresenta il cliente della palestra ed ha associato un "profilo" tariffario che influenzerà il modo in cui il costo dei biglietti di ingresso è calcolato. Il cliente mantiene anche una lista di tutti i biglietti di ingresso che ha acquistato. In questo modo sarà semplice implementare profili tariffari che prevedono offerte basate sugli acquisti pregressi. La classe immutabile *Ticket* rappresenta il biglietto ed ha una data di validità e un prezzo. Il

prezzo dovrà essere calcolato in base al profilo del cliente che lo ha acquistato. Il costruttore è privato perché le istanze di *Ticket* saranno create tramite un metodo *factory* che calcolerà il prezzo in base al profilo dell'utente. Due getter permettono l'accesso alla data per cui il biglietto è valido e al prezzo. Il metodo *toString()* utilizza *SimpleDateFormat* e *DecimalFormat* per ottenere una descrizione del biglietto formattata in modo ordinato e più leggibile.

```
package it.ioprogrammo;
import [...]
import bsh.*;
public class Ticket  {
  private final Date entranceDate;
  private final double price;
  private Ticket(Date entranceDate, double price) {
    this.entranceDate = entranceDate;
    this.price = price;
  }
  public double getPrice(){return price;}
  public Date getEntranceDate(){return entranceDate;}
  public String toString(){
    return
      "Entrance on " + new SimpleDateFormat(
            "MM/dd/yyyy").format(entranceDate) + " " +
         new DecimalFormat("€ ####.00").format(price);
  }
}
```

La classe *Customer* rappresenta un cliente della catena di palestre. L'attributo profile memorizza il nome del profilo tariffario scelto dal cliente, mentre l'attributo *tickets* memorizza tutti i biglietti precedentemente acquistati in una *List*. Il metodo più significativo è *newTicket()* invocato per ottenere il biglietto relativo al nuovo ingresso. Il codice utilizza il metodo factory *Ticket.getTicket()* al quale passa un riferimento al *Customer* per il quale si sta chiedendo il biglietto di ingresso e la data per la quale lo si richiede. Queste informazioni sono passate in modo che il metodo *factory* possa calcolare il prezzo del biglietto avendo tutte le informazioni necessarie.

```
package it.ioprogrammo;
import java.util.*;
public class Customer  {
  private List tickets = new ArrayList();
  private String profile = "default";
  public void setProfile(String profile){
    this.profile = profile;
  }
  public String getProfile(){
    return profile;
  }
  public Ticket newTicket(Date entranceDate){
    Ticket newTicket = Ticket.getTicket(this,
                                          entranceDate);
    tickets.add(newTicket);
    return newTicket;
  }
  public Ticket[] getTickets(){
    return (Ticket[])tickets.toArray(new
Ticket[tickets.size()]);
  }
}
```

IL CALCOLO DEL COSTO DEL BIGLIETTO

Il metodo *factory* per i biglietti è implementato come metodo statico della classe *Ticket*. Viene istanziato un oggetto di tipo Interpreter, che è il componente in grado di leggere ed eseguire sorgenti BeanShell. Il primo passo è quello di caricare il programma BeanShell relativo al profilo tariffario scelto dal cliente. Per instaurare questa associazione si è decisa una semplice regola di *naming*. Il file sorgente BeanShell si trova nella directory *it/ioprogrammo/* e ha come nome quello del profilo del cliente ed estensione *.bsh*. Così ad esempio per tutti i clienti con profilo tariffario uguale a "default", il sorgente BeanShell per il calcolo della tariffa si troverà nel file */it/ioprogrammo/default.bsh*. Poi sono valorizzate nell'interprete le variabili *"basePrice"* con il valore del biglietto di ingresso senza nessun particolare sconto e *"tickets"*, un array con tutti i biglietti precedentemente acquistati dal cliente. A questo punto lo script il cui path è già stato definito viene caricato ed eseguito. Si noti che grazie al passo precedente lo script potrà utilizzare le variabili *"basePrice"* e *"tickets"* anche senza valorizzarle. Al termine dell'esecuzione dello script, lanciata tramite il metodo *eval()*, il programma estrae il valore della variabile *"ticketPrice"* che lo script deve avere cura di valorizzare all'importo del biglietto proposto per il cliente. Questo prezzo viene poi assegnato al nuovo biglietto.

```
public static Ticket getTicket(
                              Customer c, Date entranceDate){
  Interpreter i = new Interpreter();
  Ticket ticket = null;
  try{
     String resPath = "/it/ioprogrammo/" + c.getProfile()
                                                 + ".bsh";
     i.set("basePrice", 20.0);
     i.set("tickets", c.getTickets());
     try{
        i.eval(new InputStreamReader(
                     Ticket.class.getResourceAsStream(
                                          resPath )));
     }catch(NullPointerException npe){
```

```
        throw new RuntimeException(
                        resPath + " not found.", npe);
    }
    double price = ((Double)i.get("ticketPrice")
                        ).doubleValue();
    ticket = new Ticket(entranceDate, price);
  } catch(EvalError ea){
    ea.printStackTrace();
  }
  return ticket;
}
```

PROFILO TARIFFARIO DI DEFAULT

Scriviamo ora il sorgente BeanShell che calcola il costo del biglietto per un cliente che ha il profilo *"default"* a cui non corrisponde nessuna applicazione di sconti e salviamolo, seguendo la regola di *naming* definita, nel file *"default.bsh"*. Questo script si limiterà a restituire nella variabile ticketPrice il valore impostato in *basePrice*.

```
ticketPrice = basePrice;
```

Aggiungiamo un metodo main alla classe *Customer* che ci permetta di simulare l'emissione di un biglietto.

```
  public static void main(String[] a){
    Customer c1 = new Customer();
    Ticket t1 = c1.newTicket(new Date());
    System.out.println(t1);
  }
```

Eseguimao il programma avendo cura di inserire nel *classpath* il *jar* di BeanShell. L'output dovrebbe essere il seguente.

```
    "Entrance on 01/20/2006  €20.00"
```

PROFILO TARIFFARIO "FREQUENT"

Implementiamo ora lo script per il profilo tariffario *"frequent"* che prevede uno sconto del 10% del prezzo se nell'ultima settimana si è acquistato un altro biglietto. Il file contenente il sorgente si troverà quindi in *"frequent.bsh"*.

```
thereIsRecentTicket(){
  oneWeekAgo = new GregorianCalendar();
  oneWeekAgo.add(Calendar.WEEK_OF_YEAR, -1);
  ticketDate = new GregorianCalendar();
  frequent = false;
  for(ticket : tickets){
    ticketDate.time=ticket.entranceDate;
    if(frequent==false){
      frequent = ticketDate.after(oneWeekAgo);
      frequent = true;
    }
  }
  return frequent;
}
ticketPrice = basePrice;
if(thereIsRecentTicket()){
  ticketPrice = ticketPrice * .8;
}
```

Alla variabile *ticketPrice* viene assegnato il prezzo base. Viene invocata la funzione *thereIsRecentTicket()* che restituisce true se nell'ultima settimana è stato acquistato qualche biglietto. In caso affermativo il prezzo del biglietto diventa l'80% di quello pieno. La funzione *thereIsRecentTicket()* utilizza un *GregorianCalendar* impostato ad una settimana fa per controllare che esista un biglietto acquistato dopo quel limite, scorrendo l'intero array di biglietti che dal codice Java abbiamo preventivamente assegnato alla variabile *"tickets"*. Si noti che nel sorgente è utilizzata per il ciclo *for* la notazione compatta *"(var : array)"* che valorizza automaticamente ad ogni giro la variabile *"var"* con uno degli elementi di *"array"*. Inoltre BeanShell mette a disposizione una notazione abbreviata per accedere alle proprietà di un oggetto normalmente accedute tramite i *getter* e i *setter*.
Ad esempio nella funzione *thereIsRecentTicket()* è possibile notare l'istruzione

```
ticketDate.time=ticket.entranceDate
```

Che equivale a

```
ticketDate.setTime(ticket.getEntranceDate());
```

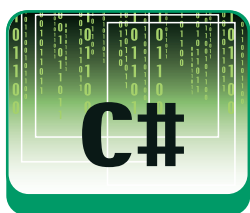
Come ultima particolarità è da notare che benché nel sorgente BeanShell venga utilizzata la classe *GregorianCalendar* nel package *java.util*, quest'ultimo non è dichiarato in nessun *import*. Questo perché le classi di alcuni package spesso utilizzati quali *java.util, java.net, java.io* e altri sono direttamente disponibili.

CONCLUSIONI

Aggiungere nuove politiche di sconto è quindi immediato utilizzando BeanShell. È difatti sufficiente inserire un nuovo script. È da notare che non è nemmeno necessario ricompilare alcunché perché ogni volta che verrà calcolato il costo lo script sarà rieseguito.

Daniele De Michelis

PRIMI PASSI CON NHIBERNATE

IMPARIAMO A USARE UN FRAMEWORK DI PERSISTENZA CHE RISOLVE IL PROBLEMA DELLA CORRISPONDENZA FRA DATI SQL, CLASSI E OGGETTI. IL NOSTRO SCOPO SARÀ RENDERE IL CODICE PIÙ MANUTENIBILE E GESTIRE I DATI IN MODO SEMPLICE

Quale programmatore non si è mai trovato in una situazione in cui il cliente ci costringe a rivedere completamente il nostro lavoro?

1. Avete sviluppato il vostro software con logica object oriented e accesso ad un database tramite il tradizionale SQL.

2. Arriva la telefonata del cliente che vi chiede di stravolgere tutto il vostro progetto, addirittura dallo schema del database.

3. Soluzione tradizionale: Modificare il database, cambiare tutte le classi, cambiare le statement SQL, ri-testare, ri-debuggare… e pregare che tutto vada per il meglio!

L'accesso ai dati è una parte fondamentale del nostro codice: centinaia se non migliaia di codice da rivedere, riscrivere, testare, magari distribuito in numerosissimi punti del vostro software, difficilmente manutenibili. Tali istruzioni sono sempre piuttosto ripetitive da scrivere e pronte ad errori di scrittura identificabili nella maggior parte dei casi solo a runtime, poiche sono interpretate dal motore di database e non dal nostro "buon" compilatore. La soluzione a questi problemi esiste…Usiamo *NHibernate!*

IL MODELLO THREE-TIER

Ovviamente esistono metodologie per scrivere applicazioni in modo molto efficiente e controllato. Uno dei migliori pattern che quotidianamente viene usato nelle applicazioni enterprise è il *three-tier* (o *three-layer*). Le applicazioni (solitamente distribuite) sviluppate secondo questa metodologia presentano tre layer applicativi i cui ruoli sono nettamente distinti: un *Presentation Layer* che contiene la logica di presentazione, un layer intermedio *(Business Layer* o *Service Layer)* che contiene tutta la logica e la descrizione degli oggetti del dominio di business, e un layer totalmente dedicato alla parte di accesso al database *(Data Access Layer),* cioè a quelle operazioni chiamate CRUD *(Create, Read, Update, Delete)* o operazioni di persistenza che si occupano del lavoro sul database. Adesso concentriamoci su quest'ultimo layer. La divisione dell'applicazione in tier isola la parte di lavoro su database in uno strato applicativo dedicato e questo è sicuramente un bene, poiché per ogni modifica implementativa sappiamo di dover andare a mettere le mani in una porzione ristretta di progetto. Ma si può e si deve andare oltre. Visto allora che le operazioni sui dati molte volte sono così ripetitive esiste un modo per automatizzarle e renderci la vita più semplice?

NASCONO GLI ORM

Il mercato software assiste così alla nascita di numerosissimi framework che si occupano di automatizzare la persistenza degli oggetti di dominio, gli *Objet Relational Mapping.* Questi tool si occupano di mappare la rappresentazione di un oggetto di dominio (una classe *.net*) in una o più righe di tabelle di database secondo il modello relazionale. La mancata corrispondenza dei due modelli di rappresentazione di dati (*OOP* e *Relazionale*) è peraltro nota nella letteratura informatica con il nome di *Object Relational Mismatch* (link). E così nasce *NHibernate:* il porting su architettura .NET del conosciutissimo *Hibernate* del mondo Java, il framework di persistenza più famoso e usato al mondo che rappresenta lo standard de facto per gli sviluppatori di applicazioni con accesso ai dati, recentemente acquisito dalla *JBoss Inc* (link). Il tutto rigorosamente OpenSource.

L'IMPORTANZA DEL DOMAIN MODEL

Uno dei guru mondiali di architetture software, *Martin Fowler,* autore peraltro di libri che sono pietre miliari nella letteratura informatica mondiale, ci da un grande suggerimento su come

affrontare e risolvere problemi di business: il *"Domain Model Pattern".* Una rete di oggetti interconnessi ciascuno dei quali ha un proprio scopo di business, contenenti contemporaneamente sia i dati che la logica di comportamento.
Esempio: un sotware bancario avrà quindi un domain model costituito da classi tipo *Cliente, ContoCorrente, Bonifico, GiroConto.* Nella progettazione di queste classi, inoltre, per sfruttare al massimo le possibilità offerte dal design orientato agli oggetti, dobbiamo utilizzare, ove possibile, caratteristiche quali ereditarietà e polimorfismo. Ci ritroveremo per esempio ad avere una classe *Bonifico* che funge da base da cui ereditano *BonificoInternazionale,* e *BonificoNazionale,* magari con comportamento polimorfico su alcune funzioni virtuali e così via. Lo studio del nostro domain model non deve essere contaminato dalla modalità con cui i nostri oggetti di dominio saranno resi persistenti sul database. Il disegno del database deve essere fatto a prescindere dal modello di dominio, poichè si baserà su un modello relazionale che manca di concetti come l'ereditarietà o il polimorfismo. E qui si incastrano perfettamente gli ORM e NHibernate in particolare, aiutando lo sviluppatore a pensare al suo software principalmente in termini di domain model, e fornendo strumenti eccellenti per quello che riguarda la parte di trasformazione di questi oggetti di dominio in righe di database. Pensare al vostro domain model in modo staccato rispetto a come questo sarà persistito migliorerà drasticamente la qualità del vostro codice. Gli ORM inoltre danno una marcia fondamentale in più. L'indipendenza dal tipo di database utilizzato. Nel caso di NHibernate basta cambiare una riga di configurazione esterna al codice per far lavorare un'applicazione che prima girava su SQL Server, in Oracle, o MySQL o tanti altri. Impareremo a liberarci delle istruzioni SQL cablate nel codice!

MA NON C'ERANO I DATASET?

La Microsoft fin dall'introduzione di .NET ha appoggiato l'uso dei suoi Dataset come ponte tra il database e i layer di business logic. Ma chi ben conosce i dataset sa perfettamente che sono terribilmente limitati per descrivere il modello di *Dominio* di una qualsiasi applicazione: non è possibile usare paradigmi OOP come polimorfismo, ereditarietà, e soprattutto siamo costretti a portarci dietro un'infrastruttura pesantissima anche per le classi più semplici. Per questo i dataset hanno avuto scarsissimo successo tra i solution architect che oggi invece hanno da sempre preferito usare delle *custom class* e delle *custom collection.*

ARCHITETTURA DI NHIBERNATE

Senza entrare troppo nel dettaglio, vediamo qual è l'architettura di NHibernate, almeno quella che serve per questi primi esempi. Questo framework si contraddistingue dal fatto che l'utente ha pochissime interfacce con cui avere a che fare, poiché il suo intervento è di tipo non invasivo. Non abbiamo la necessità di far ereditare le nostre classi da una classe base persistibile o utilizzare complicate API. Dalla **Figura 1** ci accorgiamo subito che NHibernate si colloca tra il layer applicativo e il database sottostante. La session è l'oggetto principale con cui interagiamo. Prima doverosa precisazione: non dobbiamo confondere la session di NHibernate con quella di ASP.NET. Session per NHibernate ha lo stesso significato di *Unit Of Work* (pattern definito sempre da Martin Fowler). È un contenitore in memoria di oggetti recuperati dal database che tiene anche traccia delle modifiche ad essi apportate. Si occupa inoltre della sincronizzazione di queste modifiche sul database sottostante. In pratica funge anche da cache rispetto al db. Nella vita di un'applicazione verranno istanziate e chiuse numerore session per lavorare sul database, un po come faremmo per una connection. La creazione della session è gestita da una *Session Factory,* la cui creazione dipende dalla configurazione esterna che sarà fornita (tipo di accesso al database, tipo di dialetto etc…). La *Session Factory* invece è unica per tutta la vita dell'applicazione (si può pensare ad un singleton). Come fa NHibernate a sapere come leggere, inserire, e aggiornare gli oggetti? Semplice: la configurazione è gestita tramite file *xml* esterni, uno per ogni classe. Ogni file xml contiene le informazioni con cui mappare i dati tra l'oggetto *.net* e la/le colonne delle tabelle sul database. Di solito questi file hanno estensione **.hbm.xml.*

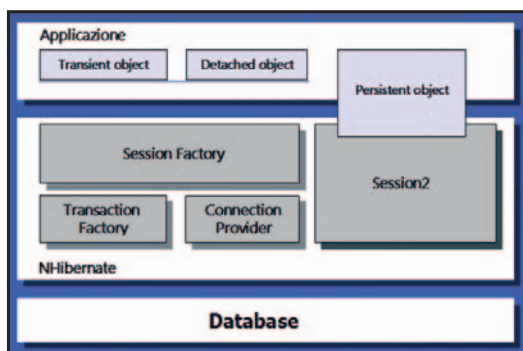
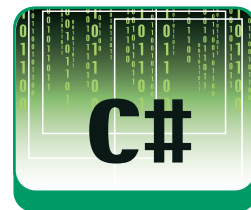
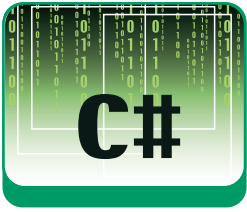



Fig. 1: ***Lo schema logico di funzionamento di NHibernate***

CONFIGURIAMO UN PROGETTO

Per l'uso di NHibernate in un qualsiasi progetto i passi da fare sono semplicissimi.

1. Scarichiamo la versione di NHibernate dal sito (*www.nhibernate.org*). Vanno bene anche solo i file binari. Allo stato attuale è presente la versione 1.0.2.
2. Creiamo un nuovo progetto di tipo *console application*.
3. Nel *solution explorer* di Visual Studio, con un clic destro sul nome del nostro project, facciamo un *Add Reference* e referenziamo questi 4 file: *NHibernate.dll, Castle.DynamicProxy.dll, Iesi.Collection.dll, HashCodeProvider.dll* (localizzateli nella directory dove avrete precedentemente installato NHibernate).
4. Creiamo un file di configurazione per l'applicazione facendo un *Add Item* e scegliendo l'*item Application Configuration*.
5. Copiamo all'interno del file di configurazione il seguente contenuto:

```
<configSections>
   <section name="NHibernate" type=
      "System.Configuration.NameValueSectionHandler,
         System, Version=1.0.3300.0,Culture=neutral,
            PublicKeyToken=b77a5c561934e089" />
</configSections>
<NHibernate>
   <add key="hibernate.connection.provider" value=
      "NHibernate.Connection.DriverConnectionProvider" />
   <add key="hibernate.dialect" value=
         "NHibernate.Dialect.MsSql2000Dialect" />
   <add key="hibernate.connection.driver_class" value=
         "NHibernate.Driver.SqlClientDriver" />
   <add key="hibernate.connection.connection_string"
         value="<<connection_string>>" />
</NHibernate>
```

6. Creiamo poi una classe singleton, il cui compito è creare la session factory, ci servirà nel resto del nostro progetto:

```
using NHibernate;
using NHibernate.Cfg;
namespace OrdiniManager
{
   public class SessionHelper
   {
      static ISessionFactory sessionFactory;
      static SessionHelper()
      {
         Configuration cfg = new Configuration();
         cfg.AddAssembly(Reflection.Assembly
                  .GetExecutingAssembly());
         sessionFactory = cfg.BuildSessionFactory();
      }
      public static ISession GetSession()
      { return sessionFactory.OpenSession();}
   }
}
```

Tutto qui. Nel file di configurazione abbiamo così introdotto una nuova section dove specifichiamo il tipo di provider di accesso ai dati, il dialetto usato secondo il tipo di database, e soprattutto la stringa di connessione al database. Per i nostri esempi utilizzeremo SQL Server 2000 (o MSDE). Per una consultazione sulle varie possibilità di configurazione potete consultare la documentazione online.
Qualche nota sulla classe *SessionHelper*. Per creare la factory, l'unico lavoro da eseguire è istanziare un oggetto *configuration* ed effettuare un *.AddAssembly()*. Questa operazione comunica a NHibernate di cercare nell'assembly specificato tutti i file **.hbm.xml* di mappatura caricandoli in memoria.

IL NOSTRO BUSINESS PROBLEM: GESTIONE ORDINI

Abbiamo configurato il nostro progetto. Adesso definiamo un problema di business da risolvere. Immaginiamo di dover sviluppare un software per la gestione di *Ordini* per una società di fornitura hardware. La **Figura 2** mostra un diagramma statico delle classi che dovremmo implementare.

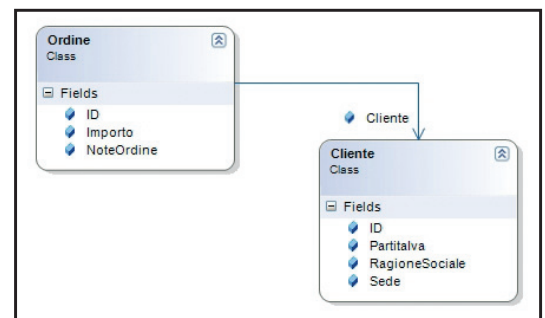



***Fig. 2:** Il diagramma statico delle classi*

In **Figura 3** ecco la tabelle che serviranno alla persistenza di questi oggetti. Su SQL Server 2000 (o successivi) creeremo quindi una tabella di nome *Ordini* con i campi mostrati in **Figura3**.

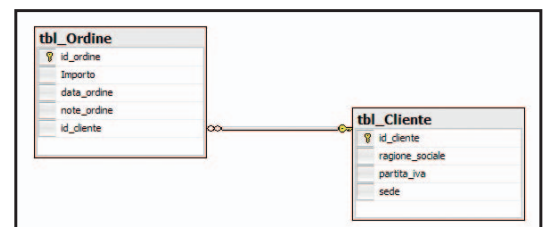
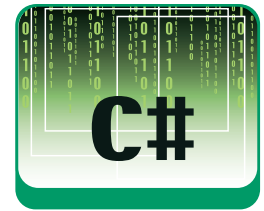



***Fig. 3:** Le tabelle del DB che serviranno per la persistenza*

Partiamo dallo studio della classe *Cliente*:

```
public class Cliente
{   public int ID = 0;
```

```
public String RagioneSociale;
public String PartitaIva;
public String PartitaIva;
}
```

Per ragioni di spazio e visto che non vi è logica particolare per l'accesso ai campi, ometteremo nelle nostre classi gli *accessors get set* (questo per rassicurare i puristi dell'*incapsulation hiding!*).

MAPPIAMO LA CLASSE CLIENTE

Come si diceva, la parte fondamentale per l'utilizzo di NHibernate è il file di mappatura. Creiamo quindi un file xml nella nostra soluzione e popoliamolo con il seguente contenuto:

```
<?xml version="1.0" encoding="utf-8" ?>
<hibernate-mapping xmlns="urn:
    NHibernate-mapping-2.0" default-access="field">
  <class name="Esempio.Cliente, Esempio" table=
                                  "tbl_Cliente">
    <id name="Id" unsaved-value="0">
      <generator class="native" />
    </id>
    <property name="RagioneSociale" name=
                           "Ragione_Sociale" />
    <property name="PartitaIva" name="Partita_Iva" />
    <property name="Sede" />
  </class>
</hibernate-mapping>
```

Per essere assistiti nell'editing dall'intellisense leggete le informazioni nel **Riquadro 1**. Esaminiamo gli elementi più importanti. Ogni mappa ha un elemento *<class>* il cui attributo name è il nome (*fully qualified*) della classe *.net*, mentre l'attributo table è il nome della tabella corrispondente sul database. Per ogni campo della nostra classe esiste un elemento *<property>* corrispondente che porta il nome della colonna (attributo *column*). Nel caso di nomi uguali column può essere omesso. Attenzione all'elemento <id>. Come per le altre property definisce la corrispondenza tra il campo id della classe e la chiave univoca della tabella, ma definisce anche altri elementi. L'elemento <generator> definisce il comportamento di NHibernate per assegnare l'id all'oggetto .net. Native vuol dire che sarà sfruttato l'id prodotto dal database server secondo il modo nativo del dialetto (il modo per recuperare l'id dell'ultimo record inserito cambia da dialetto a dialetto, ma per noi è totalmente trasparente). Importantissimo l'attributo unsaved-value che serve a far capire a NHibernate quando lanciare istruzioni di insert/update secondo il valore assegnato. Vedremo meglio più tardi. Ovviamente per una consultazione più approfondità dei numerosissimi elementi possiamo rimandare alla documentazione online (link). Il file *.hbm .xml* può essere conservato in qualsiasi posizione all'interno del progetto. L'importante è che sia impostato come risorsa embedded dell'assembly (cioè sia inglobato nella dll finale). Per fare ciò, nel solution explorer di Visual Studio, effettuate un click destro sul nome del file .hbm.xml e scegliete Property. Alla voce "Build Action" selezionate l'opzione "Embedded Resource". Approfittiamo di questa occasione per dire che ogni qual volta saranno modificati esclusivamente i files .hbm (e nessuna riga di codice applicativo), la soluzione non sarà rigenerata automaticamente. Per vedere l'effetto delle modifiche assicuratevi di forzare una build manuale prima di provare le vostre modifiche.

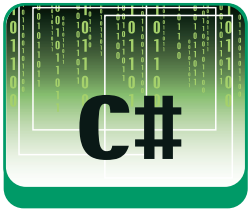
PERSISTENZA

Abbiamo configurato il progetto e creato il mapping della classe Cliente. Proviamo adesso il funzionamento di NHibernate nell'intero ciclo di vita di un oggetto. Consigliamo di controllare il flusso delle istruzioni SQL tramite il SQL Profiler come spiegato nel **Riquadro 2**. Proviamo a creare da codice un nuovo oggetto di tipo Cliente e a persisterlo sul database, il codice è davvero banale:

```
ISession session = SessionHelper.GetSession();
Cliente cliente = new Cliente();
cliente.RagioneSociale = "RagioneSociale";
cliente.PartitaIva = "0123456789012";
cliente.Sede = "Sede...";
session.SaveOrUpdate(cliente); // Viene eseguita la
                                    insert sul db.
Console.WriteLine("ID del cliente creato: " + cliente.ID);
session.Flush();
session.Dispose();
```

Subito dopo l'istruzione new Cliente(), la proprietà cliente.Id ha ovviamente valore 0. In questo stato l'oggetto cliente è definito transient. Con il metodo .SaveOrUpdate() comunichiamo a NHibernate il comando di persistere sul database l'oggetto appena creato in memoria. A questo punto entra in gioco il valore unsaved-value sulla proprietà Id. Se la proprietà cliente.Id è uguale a unsaved-value, allora NHibernate capisce che l'oggetto è stato creato in memoria e non esiste ancora sul database. Sarà quindi scatenata un'istruzione SQL di Insert per la persistenza. Ecco nel dettaglio cosa succede sul db:

```
INSERT INTO tbl_Cliente ("..") VALUES ("..");
                        SELECT Scope_Identity()
```

Oltre alla insert, notiamo nello stesso round trip

una Select Scope_Identity() che ritorna ad NHibernate l'ultimo id inserito. NHibernate completa anche il lavoro e assegna l'Id all'ordine. L'oggetto Cliente si trova adesso nello stato definito persistent. Da questo momento in poi, qualsiasi altra modifica fatta in memoria sarà sincronizzata sul database, stavolta con istruzioni di Update. Completiamo le operazioni CRUD, proviamo a recuperare un oggetto ordine già esistente sul db e cancellarlo definitivamente:

```
ISession session = SessionHelper.GetSession();
Cliente cliente = (Ordine)session.Get(typeof(Cliente), 1);
cliente.RagioneSociale = "RagionaSociale modificata";
session.SaveOrUpdate(cliente);
session.Flush();
session.Dispose();
```

Abbiamo usato session.Get() che recupera un oggetto dal database mediante chiave primaria che viene fornita nella firma del metodo. E' l'operazione più semplice, ovviamente non abbiamo avuto bisogno di specificare nomi di campi in questo caso. NHibernate sa come generare la giusta clausola where sul campo chiave primaria, poiché è tutto descritto nel file di mappatura.
Ci manca la cancellazione:

```
ISession session = SessionHelper.GetSession();
Cliente cliente = (Cliente)session.Get(typeof(Cliente), 1);
session.Delete(cliente);
session.Flush();
session.Dispose();
```

Come vedete l'unico oggetto di NHibernate con cui abbiamo avuto a che fare è sempre e solo session.
Attenzione: NHibernate per migliorare le prestazioni utilizza un metodo di scrittura tardivo chiamato transparent write behind, in pratica le istruzioni sql, saranno lanciate sul db solo saranno strettamente necessarie.
Per esempio le insert saranno lanciate subito per fare in modo che l'id ritorni subito all'applicazione, le update o le delete potranno essere posticipate, in modo da ottimizzare i round trip sul db e combinare più istruzioni in una sola. Normalmente le istruzioni pendenti saranno lanciate alla chiusura della session mediante il metodo .flush(), (o automaticamente alle commit delle transazioni che non sono oggetto di questo articolo).

PERSISTENZA DINAMICA

Ci sono tantissimi generatori di codice che prendono in pasto una tabella di un database e generano il codice c# per persistere record, e tali tool si comportano né più né meno in questo modo. Ma NHibernate è più di un generatore di codice. Proviamo ad aggiungere questi due attributi all'elemento class del file Ordine.hbm.xml, l'elemento class diventarà quindi:

```
<class name="Esempio.Cliente, Esempio" table=
                    "tbl_Cliente" dynamic-update="true"
                                 dinamic-insert="true">
```

In questo modo comunichiamo a NHibernate che quando saranno lanciate le sql di insert e update, saranno considerati solo i campi effettivamente modificati. Non dimenticate di eseguire la build manuale del progetto (abbiamo modificato solo file .hbm.xml). Rieseguendo il codice per l'inserimento e la modifica tramite profiler ci possiamo rendere conto che le INSERT e le UPDATE adesso hanno come parametri solo quelli effettivamente modificati. Questo garantisce meno traffico "inutile" sul database e soprattutto maggiore scalabilità dell'applicazione.

ASSOCIAZIONI: MANY-TO-ONE

Finora abbiamo visto un esempio semplicissimo che aveva una corrispondenza uno a uno con una tabella. Cominciamo a complicare (anche se di poco) il nostro modello. Modelliamo anche la classe Ordine. Ogni Ordine ha ovviamente un cliente che lo ha creato. Come cambia il nostro progetto? Ecco la classe Ordine:

```
public class Ordine
{
  public int ID = 0;
  public Cliente Cliente = new Cliente(); // Reference
                              ad un oggetto Cliente
  public String NoteOrdine;
  public double Importo;
}
```

Il Cliente è di tipo Cliente e non è rappresentato (come molti fanno!) con il suo id (chiave secondaria). Non dobbiamo preoccuparci di mettere id al posto delle reference fin quando stiamo sul nostro domain model. La trasformazione di reference in id al momento della persistenza è tutta demandata ad NHibernate. Creiamo un file hbm.xml per la classe Ordine

```
<?xml version="1.0" encoding="utf-8" ?>
<hibernate-mapping xmlns="urn:
    NHibernate-mapping-2.0" default-access="field">
  <class name="Esempio.Ordine, Esempio" table=
                                    "tbl_Esempio">
```

```
    <id name="Id" unsaved-value="0">
      <generator class="native" />
    </id>
    <many-to-one name="Cliente" column=
                        "IDCliente" outer-join="true" />
    <property name="DataOrdine" name=
                                    "DataOrdine" />
    <property name="Importo" name="Importo" />
  </class>
</hibernate-mapping>
```

Notiamo come novità l'elemento <many-to-one>. Questo è il primo esempio di associazione. L'attributo outer-join verrà spiegato in seguito, per il resto la definiziona è molto simile all'elemento property. Ma nasconde una grande potenza. Proviamo a crearci dei dati su cui poter lavorare, creiamo prima un oggetto Cliente, poi un Ordine:

```
ISession session = SessionHelper.GetSession();
Cliente cliente = new Cliente();
cliente.RagioneSociale = "RagioneSociale";
Ordine ordine = new Ordine();
ordine.Importo = 1000;
ordine.Cliente = cliente;
ordine.DataOrdine = DateTime.now;
session.SaveOrUpdate(ordine);
session.Flush();
session.Dispose();
```

Dal Profiler osserviamo che l'unica operazione .SaveOrUpdate() ha fatto scattare la persistenza sia per l'oggetto Cliente, che per l'oggetto Ordine, gli ID creati sul db sono stati catturati a regola d'arte e assegnati alle classi. Questa tecnica si chiama Persistence by reachability e assicura che la persistenza di un oggetto sarà estesa a tutti gli oggetti a lui collegati.
Ma abbiamo appena iniziato a divertirci.
Proviamo a fare la query di questo Ordine appena creato:

```
Ordine ordine = (Ordine) session.Get(typeof(Ordine), 1);
```

Dal profiler ci accorgiamo che la select è più complicata del solito:

```
SELECT * from ...JOIN ...
```

NHibernate ha creato una join tra le tabelle Ordine e Cliente, ha ottenuto un prodotto cartesiano, e mappato i campi sia sull'oggetto Ordine, che sull'oggetto Cliente. La corretta definizione della join è stata possibile sempre tramite i file hbm.xml. Niente più codice SQL sparso nelle nostre applicazioni e query lunghe chilometri. Inoltre questo ha permesso in un unico round trip sul server di caricare tutti i dati necessari.

STRATEGIE DI FETCHING

Scegliere se caricare i dati o meno in una select con svariate join, oppure su select differenziate, è una vera e propria strategia che dal punto di vista di scalabilità deve essere affrontata in modo serio e misurato nelle vostre applicazioni.
La potenza di NHibernate sta proprio nel fatto che mette a disposizione la possibilità di configurare questi metodi di recupero dei dati, addirittura senza toccare codice.
Approfondiamo dunque l'elemento <many-to-one>: quest'ultimo ha un attributo outer-join i cui valori validi sono true, false o auto.
Se outer-join vale true, allora NHibernate produrrà un'unica select con tante left outer join (secondo il dialetto SQL impostato) quanti saranno gli elementi <many-to-one> definiti. Se outer-join è false, le istruzioni di select saranno distinte, ma è importante dire che saranno lanciate in maniera intelligente: ogni oggetto è caricato nella cache locale della nostra session, e qual'ora siano fatte richieste successive dello stesso oggetto, sarà utilizzato l'oggetto in cache (secondo il pattern Identity Map descritto da Martin Fowler).
Avremo modo di approfondire in altre occasioni il fetching unitamente al lazy loading (caricamento tardivo) e alla cache poiché è poprio tramite questi particolari tuning che si ottengono prestazioni e vera scalabilità.

CONCLUSIONI

Abbiamo visto ancora pochissimo di NHibernate, ma speriamo di essere riusciti a far capire l'estrema potenza e flessibilità di questo strumento. Il nostro cliente ha stravolto il database? Cambiamo gli hbm.xml e le classi e lavoreremo in pieno ambiente fortemente tipizzato. Non dovremo più toccare stringhe SQL e fare test, occuparci dei caratteri di escaping, o di come si recupera l'ultimo ID inserito in SQL Server o su Oracle, o ancora su DB2. Quando ci si rende conto di dominare un framework di questo tipo non si torna più indietro. Ma attenzione: la curva di apprendimento di NHibernate (a detta stessa dei creatori) è piuttosto ripida, quindi consigliamo caldamente un periodo di 2/3 settimane di studio/apprendimento personale prima di poterlo utilizzare in progetti reali. Attenzione a metterlo in campo su progetti che hanno scadenze molto ravvicinate. Come contropartе, dopo uno sforzo iniziale NHibernate vi garantirà un livello di produttività ineguagliabile, un supporto allo sviluppo enterprise completo ed efficiente, e un miglioramento della qualità del codice e perché no anche della vita di voi sviluppatori!

Giancarlo Sudano

A TUTTA SCOMMESSA

APPASSIONATI DI TOTOCALCIO? ECCO UN ALGORITMO POSSIBILE PER OTTENERE SISTEMI DECISAMENTE AFFIDABILI SENZA SPENDERE UNA FORTUNA. IMPAREREMO MOLTO SULLA STATISTICA E CREEREMO UN ESEMPIO PRATICO...

Gli appassionati di totocalcio o di altri giochi a pronostico, si trovano sempre alle prese con il problema di individuare il miglior pronostico da giocare. Spesso si parte dal budget previsto per la giocata e quindi si individua un buon sistema di riduzione; altre volte si fa ricorso ad algoritmi di filtraggio condizionato. Ma in ogni caso, si ritorna al punto di partenza: il pronostico.
A prescindere dalla metodologia di riduzione e dagli eventi che sono presenti nella schedina, il pronostico in genere è composto da un numero ben preciso di doppie e triple, ed uno dei dubbi più frequenti è, una volta prefissata la composizione del sistema, individuare le partite sulle quali è più conveniente giocare una doppia o una tripla. Spesso risulta più agevole definire un picchetto, individuare cioè per ogni evento una probabilità di uscita per ciascun segno. Ma dopo? Come sfruttare al massimo questo picchetto? Sarebbe utile disporre di un algoritmo che, presi in pasto il nostro picchetto e la composizione del sistema in termini di doppie e triple, fornisca il pronostico che massimizza le probabilità di realizzare una vincita. Si consideri il seguente esempio in cui vi sono 4 partite e si desidera scommettere sul risultato finale *"1"*, *"X"* o *"2"* potendo disporre di una doppia e di una tripla e note le probabilità di ciascun segno:

	HOME	VISITOR	1	X	2
1	Cesena	Juventus	20	25	55
2	Inter	Lecce	40	40	20
3	Milan	Ascoli	50	30	20
4	Roma	Lazio	30	35	35

Tabella 1: Eventi e picchetto utente (in %)

1-1-1X-1X2 = 20% * 40% *(50% + 30%) * (30% + 35% + 35%) = 6,4%
1-1-12-1X2 = 20% * 40% *(50% + 20%) * (30% + 35% + 35%) = 5,6%
2-X-12-1X2 = 55% * 40% *(50% + 20%) * (30% + 35% + 35%) = 15,4%
2-1X-1-1X2 = 55% * (40% + 40%) * 50% * (30% + 35% + 35%) = 22,0%
1X2-X2-2-2 = (20% + 25% + 55%) * (40% + 20%) * 20% * 35% = 4,2%

Tabella 2: Possibili pronostici e relativa probabilità di uscita

REQUISITI

Conoscenze richieste
Basi di C#

Software
C#

Impegno

Tempo di realizzazione

Nell'esempio illustrato esistono ben 324 possibili pronostici giocabili; ciò che interessa è individuare il pronostico che presenta la maggiore possibilità di realizzazione. Più in generale il problema potrebbe essere così definito: data una serie di eventi a pronostico, tali che per ciascuno vi sia lo stesso numero di possibili risultati, siano definite per ogni risultato le probabilità di realizzazione dello stesso, in modo tale che la somma delle probabilità dei risultati di ciascun evento sia pari ad *1*; determinare il pronostico che rende massima la probabilità di realizzazione ad esso associata. Questa generalizzazione consente all'algoritmo di essere utilizzabile per diversi giochi a pronostico, come il totogol ed in genere può essere sfruttato come algoritmo decisionale per problemi di scelta opportunamente modellati.

APPROCCIO AL PROBLEMA

Per rispondere al problema, si potrebbe pensare di applicare una tecnica esaustiva, e cioè di generare tutti i possibili pronostici, calcolare la probabilità di realizzazione di ognuno e infine selezionare quello caratterizzato dalla probabilità maggiore. Tuttavia, il problema ha una naturale complessità esponenziale ed è necessario ottimizzare quanto possibile l'algoritmo per poter pervenire alla soluzione ottima in tempi accettabili. Infatti, per calcolare il numero di combinazioni che si possono sviluppare, possiamo considerare per ogni segno (i.e.: *"1"*, *"2X"*, *"21X"*, ...) il numero di combinazioni delle occorrenze desiderate rispetto ai segni disponibili; nel nostro caso, le 2 fisse possono essere posizionate sui 4 eventi in *c=4!/[2!*(4-2)!]=6* modi differenti. Ad esempio:

	HOME	VISITOR	
1	CESENA	JUVENTUS	P1
2	INTER	LECCE	P2
3	MILAN	ASCOLI	P3
4	ROMA	LAZIO	P4

Le sei combinazioni possibili per disporre le due fisse sono

P1-P2
P1-P3
P1-P4
P2-P3
P2-P4
P3-P4

Inoltre, per ogni gruppo di segni, occorre considerare il numero di combinazioni possibili a cui i vari simboli (i.e.: *"1", "X", "2"*) presi senza ordine possono dar luogo per quel segno; nel caso delle 2 fisse, esistendo 3 differenti simboli *(1-X-2)* e due posizioni, occorre considerare che le due fisse hanno un numero di segni possibili pari a *s=32=9*, arrivando così al risultato di 324 possibili pronostici per il problema di esempio.
Supposto che sia:

s=segni
C=combinazioni

Supposto che il nostro sistema si componga di due fisse, una doppia e una tripla, avremo

	Segni possibili	4 Eventi	2 Eventi	1 Eventi
2 Fisse	$s=3^2=9$ 1 1 1 X 1 2 X 1 X X X 2 2 1 2 X 2 2	C=6*9=54		
1Doppia	s=3 1-X 1-2 X-2		c=2*3=6	
1Tripla	s=1 (1X2)			c=1*1=1
Combinazioni totali				54*6*1=324

Spesso le ottimizzazioni nascono da semplici osservazioni; una prima semplice osservazione è che non ha senso considerare tutti i possibili segni, ma che in ogni evento si può considerare un solo rappresentante per ogni tipo di segno. Se prendiamo, come riferimento l'esempio iniziale, nell'evento *Cesena-Juventus,* non ha senso considerare ogni possibile fissa, ma ha senso considerare solo il segno *"2";* infatti, a parità di tutti gli altri segni, il pronostico con il segno *"2"* avrà sicuramente probabilità maggiore dei pronostici con segno *"1"* o *"X";* lo stesso discorso è valido anche per la doppia, per cui sarà sufficiente considerare la sola doppia *"2X"*. Da questa semplice osservazione, si ottiene che il peso dei segni possibili sarà sempre pari ad 1, con una notevole diminuizione di combinazioni da generare; nell'esempio, si passerà da 324 a solo 12 combinazioni da verificare. Prima di procedere con una ulteriore osservazione, è importante sottolineare che l'algoritmo non lavora sul picchetto dell'utente, ma sul picchetto associato ai segni; inoltre, nella generazione dei segni si avrà cura di preservare l'ordine decrescente delle probabilità con la convenzione di considerare il *"fattore campo",* o più genericamente l'indice crescente del simbolo, a parità di probabilità; questo in pratica afferma che un eventuale segno *"X2"* è differente da *"2X"*.

	Home	Visitor	1	x	2
1	Cesena	Juventus	20	25	55
2	Inter	Lecce	40	40	20
3	Milan	Ascoli	50	30	20
4	Roma	Lazio	30	35	35

Tabella 3: Eventi e picchetto utente (in %)

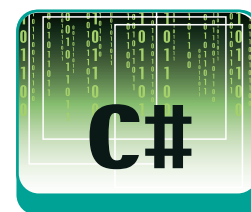
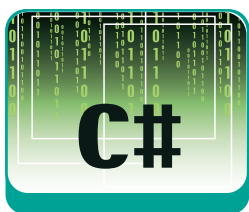
	Home	Visitor	F	D	T
1	Cesena	Juventus	2: 55	2X: 80	2X1: 100
2	Inter	Lecce	1: 40	1X: 80	1X2: 100
3	Milan	Ascoli	1: 50	1X: 80	1X2: 100
4	Roma	Lazio	X: 35	X2: 70	X21: 100

Tabella 4: Eventi e picchetto segni (in %)

A questo punto risulta facile osservare che la probabilità dell'evento certo (tripla per il totocalcio, o equivalente per altri modelli) è pari ad 1 per qualsivoglia evento. Questo permette di disinteressarsi della posizione in cui ricadono gli eventi certi (considerato ultimo segno da allocare), con una interessante ricaduta sul posizionamento del penultimo segno: essendo la probabilità, infatti, un prodotto di numeri positivi, sarà sufficiente selezionare gli *n* segni a probabilità maggiore tra quelli rimasti disponibili. Nell'esempio, se supponiamo di aver allocato le due fisse nei primi 2 eventi, sarà sufficiente selezionare il pronostico con la doppia in *evento 3* senza testare quello con la doppia sull'*evento 4,* passando così da 12 a 6 combinazioni da testare. Per capire maggiormente la qualità del risultato ottenuto, si considerino 18 eventi con 4 possibili risultati combinabili ed una composizione del sistema *6F-4D-3T-5Q:* la modalità selettiva utilizzata sul penultimo segno permette di ridurre di un fattore 56:1 il numero di iterazioni dell'algoritmo.

	18 Eventi	12 Eventi	8 Eventi
6F	18.564		
4D	3.060	495	
3T	816	220	56
Combinazioni totali			514.594.800
Combinazioni senza 3T			9.189.180

Il risultato ottenuto è già positivo, ma è ulteriormente migliorabile, partendo da una semplice considerazione: si potrebbe tentare di allocare per primi i segni che danno luogo a meno combinazioni, facendo "scivolare" alla fine i segni che incidono maggiormente sul peso computazionale dell'algoritmo e facendo crescere il peso computazionale del penultimo segno. Nell'esempio proposto, questo permetterebbe una ulteriore ridu-

zione di un fattore *8.25:1*, che nel computo complessivo porta ad una riduzione di un fattore *462:1* dell'intero algoritmo.

	18 Eventi	15 Eventi	11 Eventi
6F	18.564	5.005	462
4D	3.060	1.365	
3T	816		
Combinazioni totali			514.594.800
Combinazioni senza 6F			1.113.840

La tabella delle possibili combinazioni tiene già conto del fatto che il peso di ogni segno sia pari ad uno; infatti, considerando i 4 simboli, si sarebbe dovuto tenere in conto che ci sarebbero state 4 possibilità per la fissa, 6 per la doppia, 4 per la tripla ed una per la quadrupla con un numero totale di combinazioni dell'ordine di 10^{17}.
Nell'esempio di apertura, l'utilizzo di tutte le osservazioni prodotte porterebbe alla necessità di testare solo 4 combinazioni in luogo delle 324 possibili, provando le varie posizioni della doppia e allocando successivamente le due fisse.

	Home	Visitor	F	D	T
1	Cesena	Juventus	2: 55	2X: 80	2X1: 100
2	Inter	Lecce	1: 40	1X: 80	1X2: 100
3	Milan	Ascoli	1: 50	1X: 80	1X2: 100
4	Roma	Lazio	X: 35	X2: 70	X21: 100
Combinazioni da testare					

Tabella 5: Eventi e picchetto segni (in %)

Comb.	Posizione D	Posizione F1	Posizione F2	Probabilità
1	1	3	2	.80*.40*.50 = 16,0%
2	2	1	3	.80*.55*.50 = 22,0%
3	3	1	2	.80*.55*.40 = 17,6%
4	4	1	3	.70*.55*.50 = 19,2%
Combinazione ottima: 2 - 1X - 1 - 1X2				

LA CLASSE CMDSTACOMBINATION

L'algoritmo si basa sulla generazione esaustiva di combinazioni; sarà pertanto utile disporre di una classe che mette a disposizione delle primitive che consentano in modo agevole di generare ed iterare delle combinazioni.
La classe proposta si presta ad essere utilizzata secono lo schema classico di iterazione già presente in vari oggetti *COM:*

```
CMDSTACombination combination = new
                         CMDSTACombination();
combination.Initialize(...);
while (!combination.EOF)
{
        ...
        combination.MoveNext();
}
```

Pertanto la classe disporrà di un metodo di inizializzazione, delle proprietà per verificare lo stato dell'oggetto e il valore degli elementi della combinazione corrente, più altre proprietà quali appunto l'*EOF*. Disporrà inoltre dei metodi di avanzamento e reset dell'oggetto, e più propriamente dei metodi *MoveFirst()* e *MoveNext()*. L'oggetto istanza della classe *CMDSTACombination,* una volta inizializzato sul numero di elementi da combinare e sulla lunghezza delle combinazioni da generare, in pratica genera le combinazioni utilizzando i valori da 0 ad *n-1*, dove *n* è il numero di elementi. Questo consente alla classe di essere utilizzata in modo generico, come generatore di indici per qualsivoglia insieme di oggetti da combinare o, come più avanti nell'algoritmo, anche per combinare un insieme non consecutivo o opportunamente ordinato di interi. L'oggetto mette a disposizione anche una funzione statica che consente il calcolo delle combinazioni ottenibili, noto l'insieme degli elementi da combinare e la lunghezza della combinazione; ricordando che il numero c di combinazioni di n numeri a gruppi di m è dato da $c=n!/[m!(n-m)!]$, ovvero $c=n^*...^*(n-m+1)/[2^*...^*m]$ e che c è un numero intero, l'algoritmo effettua le moltiplicazioni al numeratore, dividendo per i valori al denominatore appena possibile, al fine di ridurre i problemi di *overflow.*

```
public static long Combinations(int numbers, int length)
{
        long tmp = 1;
        int d=2;

        for (int i = numbers; i > numbers-length; i--)
        {
            tmp *= i;
            for(;(d <= length) && ((tmp%d) == 0);)
                tmp /= d++;
        }

        return tmp;
}
```

Per quanto riguarda la generazione delle combinazioni, essendo l'*EOF* causato dalla *MoveNext(),* questa avrà il compito di testare appena invocata se la combinazione corrente è l'ultima; il test avviene in modo molto semplice sul primo elemento della combinazione: se questo è pari all'ultimo elemento assegnabile non esiste possibilità ulteriore di iterazione su ciascun elemento.
Nel caso invece di possibilità di passare alla suc-

cessiva combinazione, la procedura inizialmente trova a ritroso il primo elemento "iterarabile", e facendolo aumentare di 1 alloca ai valori successivi tutti gli elementi restanti.

```
if ((this.length == 0) || (combination[0] ==
                                    elements - length))
   eof = true;
if (EOF)
   return;
int cur = length - 1;
while ((cur >= 0) && (combination[cur] == elements
                                         - length + cur))
      cur--;
      combination[cur]++;
for (int j = cur + 1; j < length; j++)
      combination[j] = combination[cur] + j - cur;
```

LA CLASSE CMDBETSymbol

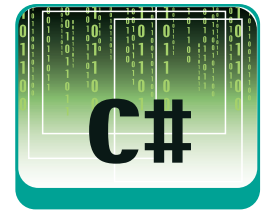
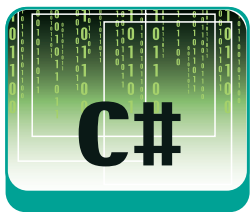
La classe *CMDBETSymbol* è una classe che mette a disposizione del programmatore un generico simbolo, ovvero un oggetto caratterizzato da un identificativo numerico, da un codice alfanumerico che rappresenta una codifica del simbolo, da una descrizione e da un peso. La classe così concepita può essere utilizzata per rappresentare sia un simbolo della schedina che un segno, associando al peso la probabilità dello stesso. Ancora, la classe può rappresentare una qualunque grandezza e infatti verrà utilizzata anche nella gestione dell'ordine dei segni da allocare, sfruttando il peso nella accezione di peso computazionale. Occorre precisare che la gestione del peso è a carico delle procedure e delle classi che istanziano oggetti della classe *CMDBETSymbol;* ad esempio, nel caso di probabilità, è importante che la somma delle probabilità dei simboli sia pari ad 1, ma il controllo è lasciato all'oggetto o procedura che fa uso della classe. La classe propone solo le proprietà necessarie alla sua gestione, quali la lettura dell'identificativo e la lettura o scrittura di tutte le altre proprietà.

DUE CLASSI PER LA GESTIONE DEGLI EVENTI

L'algoritmo di selezione del pronostico lavora sul picchetto, definito come insieme delle probabilità associate ai possibili segni da giocare; tuttavia potrebbe essere utile all'interno di una applicazione di gestione della schedina, disporre di una classe che consenta di gestire il picchetto utente, e cioè l'insieme degli eventi ed i simboli ad essi associati. La classe *CMDBETEvent* proposta utilizza un insieme di oggetti istanza della classe *CMDBETSymbol,* in cui il peso viene gestito come probabilità e si occupa, come detto in precedenza, di garantire la consistenza dei pesi. La classe, oltre alle immancabili funzioni di inizializzazione e di lettura e/o scrittura delle proprietà della classe, consente di definire il simbolo corrente (utilizzando la variabile interna *symbolIndex* e l'omonima proprietà) su cui operano metodi e proprietà caratteristiche di un simbolo. Di particolare interesse è la funzione che si occupa di garantire la consistenza dei pesi-probabilità, ovvero la funzione che garantisce che la somma dei vari pesi sia pari ad 1. Per garantire ciò, potrebbe essere necessario intervenire su alcune delle probabilità, modificandone il valore. Dovendo decidere quale simbolo modificare, si è considerata la modalità di inserimento dei dati; se ipotizziamo che tale modalità segua il verso ordinario della scrittura (da sinistra verso destra), allora possiamo ritenere "sacrificabile" il simbolo immediatamente alla destra di quello cui si assegna un dato valore e via a seguire, ovviamente considerando i simboli come sequenza circolare.

```
private void Adjust(int firstIndex)
{
   int idx = firstIndex;
   int lastIndex = (firstIndex==symbol
.GetUpperBound(0))? symbol.GetLowerBound(0)
                                       :firstIndex+1;
   double availableValue = 1;
   while (idx != lastIndex)
   {
      if (symbol[idx].Weight > availableValue)
          symbol[idx].Weight = availableValue;
          availableValue -= symbol[idx].Weight;
          idx--;
      if (idx < symbol.GetLowerBound(0))
          idx = symbol.GetUpperBound(0);
    }
    symbol[lastIndex].Weight = availableValue;
}
```

Questo metodo, privato, viene invocato dai metodi della classe che effettuano variazioni sul peso dei simboli, subito dopo le eventuali variazioni; il parametro firstIndex consente di specificare l'indice del simbolo più importante, da lasciare invariato. Di seguito a titolo di esempio si riportano la proprietà *Probability* che permette di settare la probabilità del simbolo corrente, e il metodo *SetProbability,* che consente di settare la probabilità di un simbolo senza modificare il simbolo corrente specificando l'indice del simbolo da aggiornare.

```
public void SetProbability(int symbolIndex, double val)
{
   symbol[symbolIndex].Weight = val>1?1:val;
```

```
    Adjust(symbolIndex);
}
public double Probability
{
    get { return symbol[symbolIndex].Weight; }
    set { symbol[symbolIndex].Weight =
                value>1?1:value; Adjust(symbolIndex); }
}
```

La classe *CMDBETEvent* è a sua volta utilizzata dalla classe *CMDBETEventSheet* che ha la funzione di "collettore" di eventi. Questa classe si presta quindi ad essere utilizzata come struttura dati per contenere le informazioni degli eventi in input. Si eviterà tuttavia un maggior approfondimento su questa classe, essendo la stessa di potenziale interesse per una eventuale applicazione ma di scarso rilievo per l'algoritmo trattato.

LE CLASSI PER LA GESTIONE DEL PICCHETTO

Come detto in varie occasioni, ai fini dell'algoritmo il picchetto è l'insieme delle probabilità associate non ai simboli ma ai segni. La classe *CMDBETPiquet* è strutturata in modo tale da raccogliere le informazioni relative al picchetto di un singolo evento, così come inteso ai fini dell'algoritmo.
La struttura della classe è piuttosto simile a quella proposta per la classe *CMDBETEvent,* fatta eccezione per la gestione delle modalità di valorizzazione delle probabilità dei segni. Infatti, non occorre più garantire che la somma delle probabilità sia pari ad uno, ma sarà sufficiente un controllo che il valore immesso non sia superiore a questo limite. Sulla generazione del picchetto per ogni evento è importante rispettare quanto detto sulla composizione dei segni, considerando la combinazione dei segni maggiormente probabili e preservando al loro interno l'ordinamento dei simboli rispetto alle probabilità e, in caso di pari probabilità, al "fattore campo". Analogamente a quanto visto per le classi di gestione degli eventi, viene definita una classe *CMDBETPiquetSheet* che di fatto rappresenta la struttura dati contenente il picchettò dell'intero modello. La classe *CMDBETPiquetSheet,* inoltre, offrirà le funzioni e le proprietà che consentiranno di pervenire al pronostico ottimo.

INIZIALIZZAZIONE DELLE ISTANZE

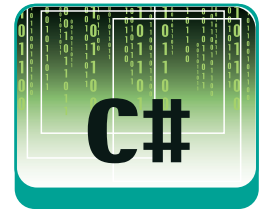
Prima di procedere con l'illustrazione delle parti più salienti dell'algoritmo, può essere utile proporre alcune porzioni di codice che utilizzano le classi fin qui presentate e che potranno essere inserite, ad esempio, nella form principale dell'applicazione. Per semplicità, saltate le parti di raccolta dei dati in input e di presentazione degli stessi in output, al fine di fornire un insieme di istruzioni che consentano l'esecuzione ed il test dell'applicazione, verranno illustrate una funzione per il riempimento casuale del "picchetto utente", la funzione di inizializzazione del picchetto vero e proprio e la funzione di calcolo della soluzione. Per comodità, l'applicazione farà uso delle seguenti variabili:

```
private CMDBETEventSheet EventSheet;
public int N_O_EVENTS = 14;
public int N_O_SYMBOLS = 3;
public int N_O_MARKS = 3;
```

L'oggetto *EventSheet,* è dunque la struttura preposta a rappresentare lo spazio dati utente, e può essere inizializzata utilizzando i valori casuali restituiti da una istanza della classe *System.Random;* una semplice funzione può prevedere il settaggio dei primi *n-1* simboli con valori compresi tra lo 0 e l'inverso del numero di simboli; l'ultimo simbolo così sarà in automatico settato dalla classe *CMDBETEvent* in modo tale da rispettare la somma ad 1 del picchetto utente. In modo euristico, possiamo prevedere un fattore correttivo di 0.4 rapportato al numero di simboli da sommare al valore casuale, per rimediare all'inevitabile sbilanciamento che altrimenti si avrebbe nei confronti dell'ultimo simbolo.

```
private void EventSheetRandomGenerate()
{
    System.Random oracle = new System.Random();

    for(int i = 0; i < N_O_EVENTS; i++)
    for (int j = 0; j < N_O_SYMBOLS - 1; j++)
    EventSheet.SetProbability(i, j, System.Math.Round(
                oracle.NextDouble() / N_O_SYMBOLS +
                        .4 / N_O_SYMBOLS, 3));

    EventSheetRefresh(); //output dati utente
}
```

È utile a questo punto disporre di una funzione che inizializzi il picchetto, così come inteso dall'algortitmo, a partire dal picchetto utente. Questa funzione dovrà per ogni evento definire l'ordine dei simboli e comporre i segni che andranno a definire il picchetto rappresentato da una istanza della classe *CMDBETPiquetSheet* passata per riferimento:

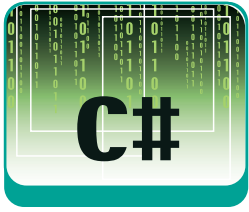
```
private bool PiquetInitialize(ref CMDBETPiquetSheet
                                    piquetSheet)
```

```
{
  String mark;
  double probability;
  int[] symbolOrder;

  symbolOrder = new int[N_O_SYMBOLS];

  int k;
  for (int i = 0; i < N_O_EVENTS; i++)
  {
    piquetSheet.SetPiquetData(
          i, String.Format("Piquet n° {0:#}", i + 1));
    for (int j=0; j < N_O_SYMBOLS; j++)
    {
      for (k = j; (k > 0) &&
                EventSheet.GetProbability(i, j) >
                    EventSheet.GetProbability(
                        i, symbolOrder[k-1]); k--)
      symbolOrder[k] = symbolOrder[k-1];
      symbolOrder[k] = j;
    }
    for (int j = 0; j < N_O_MARKS; j++)
    {
        mark = ""; probability = 0;

        for (k = 0; k <= j; k++)
        {
          mark += EventSheet.GetCode(
                  i, symbolOrder[k]);
          probability += EventSheet.GetProbability(
                  i, symbolOrder[k]);
        }
        piquetSheet.SetMarkData(i, j, mark,
          String.Format("Mark n° {0:#}", j + 1));
        piquetSheet.SetProbability(i, j, probability);
      }
  }
  k = 0;

  int marksNumber = 0;
  for (int i = 0; i < N_O_MARKS; i++)
  {
    piquetSheet.SetMarkOccurrence(
                i, System.Int32.Parse(...));
    marksNumber +=
          piquetSheet.GetMarkOccurrence(i);
  }

  return marksNumber == N_O_EVENTS;
}
```

Risulta chiaro che per ogni evento la funzione prima definisce l'ordine dei simboli sulla base delle probabilità degli stessi, e poi definisce il segno, componendo i simboli secondo l'ordine ottenuto. Dopo aver composto tutti i segni e calcolato per ciascuno di essi la probabilità, la procedura definisce la struttura in termini di occorrenza dei segni nel pronostico che si vuole calcolare; la procedura infine, termina correttamente se il numero totale di segni richiesto coincide con il numero di eventi in gioco. Un ulteriore controllo potrebbe essere aggiunto per garantire che il picchetto soddisfi il vincolo che la probabilità associata all'ultimo segno sia uguale ad 1. Per concludere, è utile presentare la procedura che invoca il calcolo del picchetto ottimo; per comodità supponiamo che il risultato sia inserito in un oggetto di tipo *ListView lvResult* impostato con modalità di visualizzazione *List*.

```
private void GetOptimalCard()
{
  CMDBETPiquetSheet PiquetSheet = new
                        CMDBETPiquetSheet();
  PiquetSheet.Initialize(N_O_EVENTS, N_O_MARKS);

  if (PiquetInitialize(ref PiquetSheet))
  {
      PiquetSheet.OptimalCardRefresh();

      lvResult.Items.Clear();
      System.Windows.Forms.ListViewItem lvi;
      for(int i=0; i < N_O_EVENTS; i++)
        lvi = lvResult.Items.Add(
                PiquetSheet.GetOptimalMark(i));
        lvi = lvResult.Items.Add(
            String.Format("{0:##0.#######}%",
            PiquetSheet.OptimalProbability * 100));
  }
  else
  {
      lvResult.Items.Clear();
      lvResult.Items.Add("Verificare parametri");
  }
}
```

OptimalCardRefresh

La classe *CMDBETPiquetSheet*, come detto in precedenza, consente non solo la definizione del picchetto secondo i criteri dell'algoritmo, ma anche di calcolare il pronostico ottimo in termini di probabilità.
Il calcolo avviene in due fasi; inizialmente avviene la preparazione delle strutture dati sulla base dei parametri dell'istanza, quindi viene invocata la procedura che effettua la scansione dei possibili pronostici alla ricerca di quello ottimo. La procedura di scansione dei pronostici, com'era facile aspettarsi, è una procedura ricorsiva.
Il metodo Initialize viene invocato per inizializzare le variabili private della classe *CMDBETPiquetSheet:*

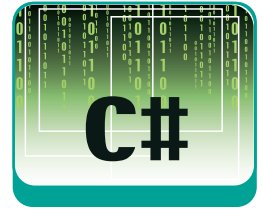
```
private CMDBETPiquet[] Piquet;
```

```
private int piquetIndex = 0;
private int markIndex = 0;

private int marksNumber = 0;
private int[] markOccurrence;
private CMDBETSymbol[] markOrder;

private int[] optimalCard;
private double optimalProbability;

private int[] currOptimalCard;
private double currOptimalProbability;

public void Initialize(int piquetsNumber, int
                                        marksNumber)
{
    int i;

    Piquet = new CMDBETPiquet[piquetsNumber];
    for (i = Piquet.GetLowerBound(0); i <=
                      Piquet.GetUpperBound(0); i++)

        Piquet[i] = new CMDBETPiquet(i+i, "" ,
                                       marksNumber);
        PiquetIndex = 0; MarkIndex = 0;

        markOccurrence = new int[marksNumber];
        optimalCard = new int[piquetsNumber];
        this.marksNumber = marksNumber;
}
```

La procedura *OptimalCardRefresh()* prepara i dati prima della invocazione della procedura ricorsiva *OptimalCardRetrieve(...)*, che richiede in input l'indice del segno da allocare e la lista degli eventi disponibili per l'allocazione.
In particolare, la procedura definisce l'ordine di allocazione dei segni, utilizzando come peso il numero di combinazioni possibili in accordo alla composizione del sistema. Per comodità, verrà utilizzato come peso il numero di combinazioni ottenibili sulla totalità degli eventi senza procedere come nell'esempio iniziale calcolando il numero esatto delle combinazioni ottenibile man mano che viene determinato il segno da allocare; questa scelta di comodo di fatto non altera le considerazioni fatte in precedenza, in quanto non verrà mai utilizzato il numero di combinazioni, ma soltanto l'ordinamento in base al peso computazionale.

```
public void OptimalCardRefresh()
{
   int eventsNumber = Piquet.GetUpperBound(0)
                     - Piquet.GetLowerBound(0) + 1;

   markOrder = new CMDBETSymbol[marksNumber];
   CMDBETSymbol tmpMark;
   long combs;
   for (int i=0; i < marksNumber - 1; i++)
   {
      combs = CMDSTACombination.Combinations(
                   eventsNumber, markOccurrence[i]);
      int k = i;
      markOrder[k] = new CMDBETSymbol(i, "", "",
                                             combs);
      while ((k > 0) && (markOrder[k].Weight
                     < markOrder[k-1].Weight))
      {
         tmpMark = markOrder[k];
         markOrder[k] = markOrder[k-1];
         markOrder[k-1] = tmpMark;
         k--;
      }
   }
   markOrder[marksNumber - 1] = new
        CMDBETSymbol(marksNumber - 1, "", "", -1);

   optimalProbability =  0;
   currOptimalCard = new int[eventsNumber];
   currOptimalProbability =  1;
   int[] availableList = new int[eventsNumber];
   for (int i = 0; i < eventsNumber; i++)
      availableList[i] = i;
   OptimalCardRetrieve(0, ref availableList);
}
```

OptimalCardRetrieve

La funzione *OptimalCardRetrieve* è il nucleo centrale dell'intero algoritmo; è una procedura ricorsiva che ha il compito di iterare tutte le possibili combinazioni per i primi *n-2* segni, per poi procedere come descritto inizialmente nella allocazione degli ultimi 2 segni. La procedura pertanto presenta al suo interno due blocchi di istruzioni, uno per la clausola di uscita dalla ricorsione, e l'altro per la generica iterazione che corrisponde alla ricerca esaustiva sull'*i-esimo* segno.
La clausola di uscita, come detto, si presenta quando il segno da allocare è il penultimo segno; lo scheletro della funzione pertanto sarà il seguente:

```
private void OptimalCardRetrieve(int marksCursor, ref
                                   int[] availableList)
{
   double startingOptimalProbability =
                            currOptimalProbability;
   int availablesNumber =
                   availableList.GetUpperBound(0) -
                availableList.GetLowerBound(0) + 1;
   if (marksCursor == marksNumber - 2)
   {
      ... // allocazione ultimi 2 segni
   }
   else
```

```
  {
    ... // allocazione i-esimo segno
    ... // costruzione availableNextList

    OptimalCardRetrieve(marksCursor + 1, ref
                                availableNextList);
  }
}
```

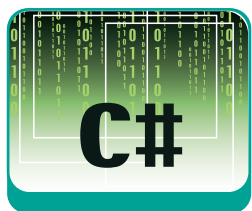
L'allocazione degli ultimi 2 segni, come detto, prevede l'ordinamento decrescente per probabilità degli eventi disponibili: di questi, i primi *k* verranno associati al penultimo segno, e i restanti all'ultimo segno (*k* è il numero di occorrenze richieste per il penultimo segno). Ovviamente, si parla di i-esimo segno con riferimento all'ordine dei segni ottenuto sul peso computazionale. Dopo aver effettuato l'allocazione, e contestualmente aver aggiornato la probabilità del pronostico corrente, il risultato ottenuto viene confrontato con l'ottimo attuale e se migliore viene memorizzato come ottimo corrente.

```
int[] bestOthers = new int[availablesNumber];
int k = 0;
for (int i=0; i < availablesNumber; i ++)
{
  k = i;
  while ((k > 0) && (Piquet[availableList[i]]
   .GetProbability(markOrder[marksNumber-2 ].Id) >
           Piquet[bestOthers[k-1]].GetProbability(
                    markOrder[marksNumber-2].Id)))
  {
    bestOthers[k] = bestOthers[k-1];
    k--;
  }
  bestOthers[k] = availableList[i];
}

for (int i=0; i < markOccurrence[
           markOrder[marksNumber-2].Id]; i++)
{
  currOptimalCard[bestOthers[i]] = markOrder[
                             marksNumber-2].Id;
  currOptimalProbability *= Piquet[bestOthers[i]]
    .GetProbability(markOrder[marksNumber-2].Id);
}

for (int i=0; i < markOccurrence[markOrder[
                       marksNumber-1].Id]; i++)
 currOptimalCard[bestOthers[markOccurrence[
          markOrder[marksNumber-2].Id] + i]] =
                              marksNumber-1;

if (currOptimalProbability > optimalProbability)
{
  optimalProbability = currOptimalProbability;
  for (int i = optimalCard.GetLowerBound(0); i
                <= optimalCard.GetUpperBound(0); i++)
  optimalCard[i] = currOptimalCard[i];
}
```

La parte ricorsiva della procedura invece istanzia un oggetto combination della classe *CMDSTA-Combination* per l'iterazione delle combinazioni; in caso di proprietà *EOF* dell'oggetto *combination* (*i.e.*: se il numero di occorrenze per il segno corrente fosse pari a zero) pari a *TRUE*, la procedura procede alla ricorsione sul segno successivo, conseguendo come lista di valori ammissibili la lista corrente:

```
CMDSTACombination combination =
                        new CMDSTACombination();
int occurrence = markOccurrence[
                     markOrder[marksCursor].Id];
combination.Initialize(availablesNumber, occurrence);
if (combination.EOF)
  OptimalCardRetrieve(marksCursor + 1,
                             ref availableList);
else
{
  while (!combination.EOF)
  {
    ... // allocazione segni e chiamata ricorsiva
    combination.MoveNext();
  }
}
```

La parte di allocazione dei segni e di preparazione e chiamata ricorsiva riparte dal valore corrente di probabilità del pronostico memorizzata ad ingresso funzione e iterando sui numeri presenti nella lista degli eventi disponibili, ne verifica l'appartenenza o meno alla combinazione corrente: se il numero appartiene alla combinazione, nel pronostico corrente viene memorizzato il segno selezionato per l'evento e l'evento non viene incluso nella lista da passare alla chiamata successiva, altrimenti l'evento entra nella prossima lista di eventi disponibili e il valore del pronostico per l'evento viene posto a *-1* (utile più che altro in fase di debug, ma non necessario per il corretto funzionamento dell'algoritmo). Una volta completata la scansione sugli eventi disponibili è possibile effettuare la chiamata ricorsiva.

```
currOptimalProbability = startingOptimalProbability;
int[] availableNextList = new int[availablesNumber
                                  - occurrence];
int k = 0, j =0;
for (int i = 0; i < availablesNumber; i++)
if ((k < combination.Length) &&
                  (combination.Index(k) == i))
{
    currOptimalCard[availableList[i]] =
```

```
                    markOrder[marksCursor].Id;
    currOptimalProbability *= Piquet[availableList[i]]
        .GetProbability(markOrder[marksCursor].Id);
    k++;
}
else
{
    currOptimalCard[availableList[i]] = -1;
    availableNextList[j] = availableList[i];
    j++;
}
OptimalCardRetrieve(marksCursor + 1,
                    ref availableNextList);
```

Una ulteriore ottimizzazione può essere effettuata introducendo una tecnica di *cut* nella generazione delle combinazioni; infatti, poiché la probabilità del pronostico corrente è un valore che dovrà essere moltiplicato per le probabilità dei prossimi segni individuati, ed essendo tutti questi numeri inferiori all'unità, non ha senso proseguire con la generazione del pronostico corrente quando la sua probabilità non può essere migliore del valore dell'ottimo attuale. Il taglio potrebbe risultare tanto più efficace quanto prima si raggiunge un valore alto di ottimo attuale; a tal proposito potrebbe essere introdotta l'euristica di ordinare la lista degli eventi disponibili per probabilità decrescente. Ovviamente la tecnica di cut introduce dei costi computazionali (ordinamento delle liste di eventi, operazioni di confronto) e non sempre sortisce i risultati voluti; tuttavia gli eventuali effetti positivi di una tecnica di cut in genere permettono una notevole riduzione delle iterazioni per algoritmi di questo tipo e il loro utilizzo può essere lasciato alla scelta del programmatore o ancora a valutazioni euristiche fatte dallo stesso algoritmo. Per completezza, si riportano le porzioni di codice che implementano l'euristica, ed in particolare, la generazione della lista nella *OptimalCardRefresh()*, la generazione della *avaliableNextList* nel ramo ricorsivo della *OtpimalCardRetrieve()* e l'allocazione degli ultimi due segni nella clausola di uscita della stessa procedura, che non richiede più la parte di ordinamento essendo la lista già ordinata.

```
for (int i = 0; i < eventsNumber; i++)
{
  int k = i;
  while ((k > 0) && (Piquet[i].GetProbability(
              markOrder[0].Id) > Piquet[availableList[
               k-1]].GetProbability(markOrder[0].Id)))
  {
    availableList[k] = availableList[k-1];
    k--;
  }
  availableList[k] = i;
}

//availableNextList[j] = availableList[i];
int w = j;
while ((w > 0) && (Piquet[availableList[i]]
        .GetProbability(markOrder[marksCursor+1].Id)
        > Piquet[availableNextList[w-1]].GetProbability(
                  markOrder[marksCursor+1].Id)))
{
  availableNextList[w] = availableNextList[w-1];
  w--;
}
availableNextList[w] = availableList[i];
j++;

for (int i=0; i < markOccurrence[
                markOrder[marksNumber-2].Id]; i++)
{
  currOptimalCard[availableList[i]] = markOrder[
                          marksNumber-2].Id;
  currOptimalProbability *= Piquet[availableList[i]]
      .GetProbability(markOrder[marksNumber-2].Id);
}
for (int i=0; i < markOccurrence[markOrder[
                     marksNumber-1].Id]; i++)
currOptimalCard[availableList[markOccurrence[
              markOrder[marksNumber-2].Id] + i]] =
                               marksNumber-1;
```

In particolare, non occorre variare i controlli sulla combinazione corrente, in quando questa viene fatta sfruttando la natura di indice degli elementi dell'oggetto *combination*.

ESEMPIO

Con riferimento alla istanza presa in esame in occasione delle considerazioni sulla complessità, cioè al modello composto da 18 eventi con la compo-

	A	B	C	D
1	140	190	195	475
2	205	200	70	525
3	250	355	95	300
4	505	110	55	330
5	175	520	180	125
6	145	535	150	170
7	270	185	400	145
8	465	405	80	50
9	445	115	120	320
10	135	210	395	260
11	285	65	295	355
12	490	75	275	160
13	290	220	225	265
14	310	315	215	160
15	75	530	385	10
16	335	90	100	475
17	235	240	365	160
18	265	80	355	300

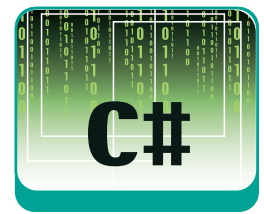
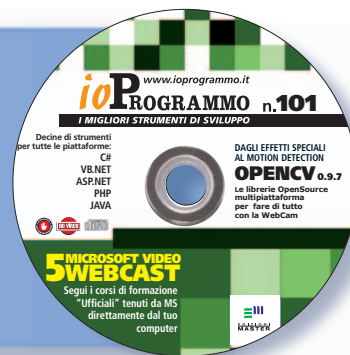
sizione *6F-4D-3T-5Q* per il risultato finale, consideriamo il picchetto utente proposto (riferito a 1000): A questo corrisponderà il picchetto, inteso ai fini dell'algoritmo, seguente:

1	D	DC	DCB	DCBA
	475	670	860	1.000
2	D	DA	DAB	DABC
	525	730	930	1.000
3	B	BD	BDA	BDAC
	355	655	905	1.000
4	A	AD	ADB	ADBC
	505	835	945	1.000
5	B	BC	BCA	BCAD
	520	700	875	1.000
6	B	BD	BDC	BDCA
	535	705	855	1.000
7	C	CA	CAB	CABD
	400	670	855	1.000
8	A	AB	ABC	ABCD
	465	870	950	1.000
9	A	AD	ADC	ADCB
	445	765	885	1.000
10	C	CD	CDB	CDBA
	395	655	865	1.000
11	D	DC	DCA	DCAB
	355	650	935	1.000
12	A	AC	ACD	ACDB
	490	765	925	1.000
13	A	AD	ADC	ADCB
	290	555	780	1.000
14	B	BA	BAC	BACD
	315	625	840	1.000
15	B	BC	BCA	BCAD
	530	915	990	1.000
16	D	DA	DAC	DACB
	475	810	910	1.000
17	C	CB	CBA	CBAD
	365	605	840	1.000
18	C	CD	CDA	CDAB
	355	655	920	1.000

In rosso sono riportati i segni in corrispondenza dei quali si ottiene l'ottimo, mentre in azzurro ci sono i riportati i valori più alti per ogni segno. Il valore di ottimo presenta lo 0,66% di probabilità di realizzazione; questo risultato, contrariamente alle apparenze, non è un risultato negativo: si pensi che la probabilità di fare un ambo secco su una ruota è pari allo 0,25%!!! Ovviamente, la probabilità di fare risultato è fortemente vincolata alla qualità del picchetto. Nella particolare istanza proposta, la tecnica di *cut* non ha sortito alcun effetto, in quanto il valore minimo di probabilità per il taglio sul ramo riscorsivo si è fermato allo 0,86% e non si è ritenuto opportuno agire sulla allocazione degli ultimi due segni, in quanto operazione lineare. Tuttavia occorre notare che gli ultimi due segni rappresentavano ben 11 eventi su un totale di 18 (quasi 2/3) e questo è sufficiente a spiegare il mancato taglio da parte dell'euristica. È però importante osservare che nella classe dei problemi a complessità esponenziale, è in genere opportuno apportare ogni possibile miglioramento si intraveda. Nell'esempio proposto, su una macchina Pentium D 3GHz, 1 GB RAM, l'algoritmo ha fornito la soluzione in circa 4 secondi; la generazione esaustiva avrebbe richiesto, quindi, approssimativamente 30 minuti; infine sarebbe stato inutilizzabile l'algoritmo che avesse generato tutte le combinazioni su tutti i possibili segni.

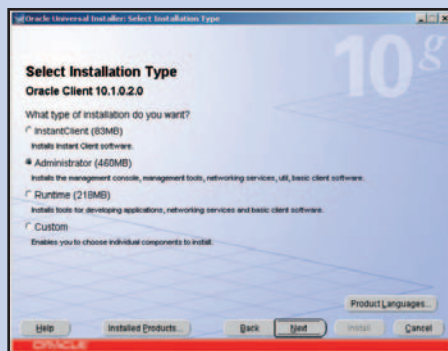
CONCLUSIONI

L'algoritmo ora è pronto per essere testato e utilizzato; tuttavia è necessario fare alcune precisazioni, in particolare sulla reale possibilità di aumentare le probabilità di vincita. Da un punto di vista matematico l'algoritmo in effetti permette di ottenere il pronostico che ha maggiori possibilità di riuscita, ma questo a condizione che il picchetto dell'utente costituisca un valido modello della realtà. Se utilizzassimo ad esempio le quote dei bookmaker per ottenere il picchetto, sicuramente escluderemmo la possibilità di considerare nel pronostico finale le cosiddette "sorprese". Infatti, sempre in riferimento ad una ipotetica applicazione per il totocalcio, se immaginiamo che la prima in classifica incontri l'ultima in classifica, sicuramente il picchetto sarebbe molto a favore della prima in classifica e se alla vittoria di questa è associata una probabilità alta difficilmente il pronostico conterrà un segno diverso dalla fissa; tuttavia, uno scommettitore particolarmente abile o fortunato, potrebbe intravedere in questa partita la possibilità di una sorpresa e decidere di associare all'evento un picchetto che dia maggiori possibilità al pareggio o, perché no, alla sconfitta della prima in classifica. In questo caso, l'ingresso della "sorpresa" nel pronostico sarebbe resa possibile dal picchetto e vincolata esclusivamente dai picchetti delle altre partite. Ovviamente le limitazioni di uso nell'ambito dei giochi a pronostico dipendono solo dalla fantasia del giocatore; si potrebbe pensare, ad esempio, di utilizzare l'algoritmo per decidere la giocata da effettuare in un gruppo di scommettitori. Ogni componente del gruppo potrebbe indicare un proprio picchetto (o un pronostico da convertire in picchetto); facendo girare l'algoritmo sul picchetto ottenuto come media di tutti i picchetti, l'applicazione potrebbe decidere il pronostico più indicato sulla base delle considerazioni di tutti i componenti del gruppo di scommessa. Da notare, infine, che l'algoritmo si presta notevolmente ai giochi a pronostico, ma è a tutti gli effetti un algoritmo decisionale esatto, e pertanto è applicabile a qualunque problema decisionale modellabile secondo lo schema generale del problema ad esso associato.

Carmelo Durante

SOFTWARE SUL CD

Oracle 10g Express Edition

PER LA PRIMA VOLTA IN VERSIONE GRATUITA, ARRIVA IL DB PER LE AZIENDE

Oracle si è sempre caratterizzata per la sua volontà di servire il mercato B2B, ed è ancora così. Per anni il suo database è stato il leader nel segmento aziendale, ed ancora oggi detiene un'enorme fetta di questo mercato. Fino a ieri nessuna o quasi nessuna applicazione destinata alle aziende medio piccole ha potuto usufruire della potenza di questo DB soprattutto in relazione agli enormi costi da sostenere. Oggi tutto è cambiato e Oracle per la prima volta presenta una versione Express Gratuita del proprio strumento principe. Si tratta di una novità non da poco. Questa versione è identica in tutto e per tutto alla versione standard eccetto che per il fatto di poter servire un solo processore, gestire fino a un massimo di 4GB di dati, e utilizzare un solo GB di RAM. I linguaggi con cui può essere integrato tramite appositi connector sono i più svariati: Java, C, PHP, la piattaforma .NET. Si tratta di una vera e propria rivoluzione a

tutto vantaggio degli sviluppatori che possono in questo modo affiancare ai tradizionali Microsoft SQL server, MySQL, PostgreSQL un nuovo strumento la cui affidabilità è proverbiale. Di fatto la versione Express di Oracle 10G può essere liberamente usata in tutta una serie di contesti, redistribuito con le proprie applicazioni e installato sui server in modo multiutente. Sicuramente una novità eccezionale da prendere con entusiasmo. Vi invitiamo a provare subito questo eccezionale strumento, e apprenderne ulteriori caratteristiche leggendo il bell'articolo di Fabrizio Fortino in questo stesso numero di ioProgrammo

AXIS 1.3

IL FRAMEWORK PER LO SVILUPPO DI WEB SERVICE IN JAVA

Che i Web Service rappresentino una risorsa enorme per la programmazione è ormai noto, in questo stesso numero di ioProgrammo diamo enorme spazio all'argomento proprio con una serie di esempi pratici per lo sviluppo. Mancano dai nostri esempi proprio quelli relativi a Java di cui ci occuperemo nei numeri successivi della rivista. Axis è comunque il tool fondamentale per poter sviluppare WS con il linguaggio di SUN. Il suo utilizzo è abbastanza semplice, e supporta tutte le caratteristiche avanzate come ad esempio la generazione dinamica dei WSDL oppure il debugging delle trasmissioni SOAP. Uno strumento indispensabile di cui non mancheremo di parlare ulteriormente

Directory: /Axis

BEANSHELL 2.0B4

PER DOTARE LE TUE APPLICAZIONI JAVA DI UN LINGUAGGIO DI SCRIPTING

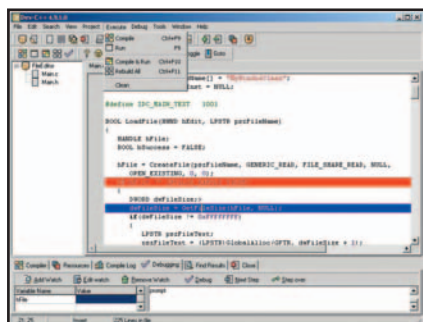
Ce ne parla approfonditamente Federico Paparoni nel bell'articolo presente in questo stesso numero di ioProgrammo. Beanshell è un linguaggio di scripting e al contempo un engine embedded per le tue applicazioni Java. Sostanzialmente si può inserire l'engine all'interno dell'applicazione e consentire agli utenti di utilizzare il linguaggio per estendere il software con moduli. In pratica una sorta di linguaggio Macro applicabile direttamente al software sviluppato da te. Molto interessante e utile in tutti quei casi in cui si vuole offrire una profonda personalizzazione senza dover ogni volta ricompilare il codice

Directory:/ BeanShell

DEV CPP 4.9.9.2

UN EDITOR C++ A BASSO COSTO

Dev C++ è un editor distribuito su licenza GPL, come tale non ha costi relativi al diritto d'autore. Come tutto

o quasi tutto il software GPL la sua economicità non è affatto sinonimo di scarsa qualità. Al contrartio Dev C++ è uno degli editori più amati ed utilizzati da chi sviluppa in C++.

Le caratteristiche sono notevoli. Si va dal Debugger integrato, al project Manager, al Class Browser, al Code Completion. L'insieme di queste caratteristiche, oltre una leggerezza innata dell'ambiente lo rende particolarmente comodo da utilizzare per sviluppare progetti C++ anche di grandi dimensioni

Directory:/ Dev CPP

ECLIPSE 3.1.2

L'IDE DI PROGRAMMAZIONE UNIVERSALE

Eclipse è diventato in breve tempo uno dei prodotti più usati dai programmatori di tutto il mondo. Si tratta ovviamente di un ambiente di programmazione. La prima caratteristica da mettere in evidenza è che l'intero eclipse è scritto in Java e questo lo rende disponibile con funzionalità omogenee sia su Linux che su Windows.

La seconda caratteristica è relativa alla sua architettura a plugin. Se da un lato l'ambiente si presenta di default come destinato alla programmazione Java, con semplici plugin diventa un IDE per PHP, per C++ o per praticamente qualunque altro linguaggio disponibile. Inoltre sempre grazie alla sua architettura a plugin, è possibile aggiungere funzionalità aggiuntive che lo rendono ad esempio un ambiente RAD e Visual nel caso si installi il plugin Visual Eclipse.

Directory: /DevCPP

J2SE 1.5.0.6

IL FRAMEWORK ESSENZIALE PER PROGRAMMARE IN JAVA

Se siete dei programmatori java non potete prescindere dall'installare il JDK, si tratta del framework di base che contiene il compilatore, le utility e le librerie per potere iniziare a lavorare con Java. Quello che vi presentiamo è il secondo aggiornamento della versione 1.5 rilasciato da brevissimo tempo e contenente alcune migliorie dal punto di vista della security e moltissimi Bug Fix che migliorano in linea generale l'usabilità e l'affidabilità del linguaggio

Directory:/ J2SE

JAVA LAUNCHER 2.0

LANCIARE APPLICAZIONI JAVA CON UN DOPPIO CLIC

Un tool di rara semplicità che consente di distribuire applicazioni Java, garantendo all'utente la massima semplicità nell'avvio. Java Launcher comprime tutte le classi e le risorse relative ad un'applicazione in un unico file .EXE che potrà essere lanciato come una comune applicazione Windows. Gratuito.

Directory:/ javalauncher

NUSOAP 0.6.3

LA LIBRERIA PER SVILUPPARE WEB SERVICES CON PHP

Ne abbiamo parlato abbondantemente negli esempi legati allo sviluppo di WS con Php in questo stesso numero di ioProgrammo. Si tratta di una libreria di veramente pochi Kb di dimensione, ma straordinariamente completa. Consente la generazione automatica del WSDL, il debugging della trasmissione dei messaggi SOAP ed esporta metodi e interfacce abbastanza semplici da rendere la programmazione dei WS estremamente rapida. Viene ancora oggi preferita all'estensione "ufficiale" presente in PHP5 perché nonostante sia ancora leggermente più lenta di essa tuttavia offre alcune caratteristiche che la nuova versione non offre, come ad esempio proprio la generazione automatica del WSDL

Directory:/ Nusoap

VS.PHP For Visual Studio 2005

L'ADDIN PER USARE VISUAL STUDIO COME EDITOR PER PHP

Si tratta di una vera e propria novità. Utilizzare uno degli editor più evoluti al mondo per programmare per il web utilizzando il linguaggio PHP. Alcuni potrebbero pensare che si tratti in fondo del solito editor con sintassi colorata, e invece tutte le funzioni avanzata ci sono tutte, dalla syntax highlighting alla code complexion, più alcune funzionalità decisamente avanzate che non si trovano tipicamente nemmeno in editor nativi per

PHP come ad esempio il supporto a smarty il template designer di PHP utilissimo per separare la logica di programmazione da quella di design. Si tratta insomma di uno strumento di eccezionale rilevanza che può diventare realmente significativo per coloro che fino ad ora non hanno trovato un IDE sufficientemente evoluto per lo sviluppo delle proprie Web Application.

MYSQL 5.0

LO SCHELETRO DI INTERNET

Così tanti sono i progetti su internet che fanno capo a MySQL che si può tranquillamente dire che MySQL è una delle strutture portanti del Web. Nato come DB ultraleggero per servire progetti di piccole dimensioni si è evoluto nel tempo fino a diventare uno dei database con il maggior numero di funzionalità esposte. 0

Si va dalla ricerca full text al supporto alle transazioni senza per questo dimenticare la leggerezza del sistema. MySQL è estremamente semplice da usare e al contempo estremamente potente e questo ne fa uno dei server di database più usati al mondo. Qualcuno ancora lo taccia di non essere sufficientemente professionale da reggere il confronto con i DB commerciali più costosi, noi non siamo fra questi. MySQL è del tutto paragonabile a server di DB del costo di svariate migliaia di euro e in molti casi prevale in prestazioni e affidabilità.

Directory: /Mysql

NHIBERNATE 1.0.2

IL TOOL DI PERSISTENZA DEI DATI PER ECCELLENZA

Ce ne parla Giancarlo Sudano in questo stesso numero di ioProgrammo. Nhibernate è il porting del famoso Hibernate per Java su piattaforma .NET.

La sua utilità è estrema, consente infatti di trasformare le normali Query SQL in oggetti e classi di modo che siano omogenei con il codice a cui uno sviluppatore è abituato. Oltre a questo Nhibernate è un tool di persistenza che consente di “salvare” lo stato dei dati su uno storage, anche questo in modo indipendente dal linguaggio SQL. A fare da ponte fra l'architettura del database e il linguaggio ad oggetti sarà un semplice file XML.

Directory:/Nhibernate

PYTHON 2.4.2

IL LINGUAGGIO EMERGENTE

Di Python ci siamo occupati a più riprese su ioProgrammo nel corso del tempo e torneremo nuovamente a farlo. Si tratta di un linguaggio realmente interessante che offre agli sviluppatori una serie di caratteristiche innovative che lo rendono piuttosto flessibile. Intanto si tratta di un linguaggio ad oggetti fra quelli che interpretano la programmazione ad oggetti nella maniera più standard possibile.

Si tratta poi di un linguaggio interpretato, modulare e multipiattaforma. Può essere esteso con librerie esterne, può funzionare su Windows e su Linux senza modifiche di rilievo eccetto nel caso in cui il software debba interagire in modo stretto con il sistema. Proprio per quanto riguarda linux di fatto Python viene utilizzato dalla maggior parte delle distribuzioni come base per la costruzione dei propri tool per la gestione del sistema. In ambiente Windows si sta rapidamente affermando anche per la costruzione di applicazioni dotate di interfacce grafiche e questo testimonia la flessibilità del linguaggio

Directory:/ Python

PHP 5.1.2

IL LINGUAGGIO PRINCIPE PER IL WEB

PHP è decisamente sempre più il linguaggio che consente di sviluppare applicazioni Web in modo rapido ed efficiente. La release che vi presentiamo in questo numero è la 5.1.2 che corregge alcuni bug di sicurezza relativi alla precedente. La forza di PHP risiede nella curva di apprendimento sostanzialmente nulla, nella completezza del linguaggio, nella capacità di essere da un lato un linguaggio procedurale atipico in quanto probabilmente l'unico sopravvissuto fra i linguaggi procedurali, ma anche un grande linguaggio ad oggetti, per quanto ed ancora più stranamente la migrazione alla programmazione ad oggetti che in altri settori è stata quasi istantanea per quanto riguarda PHP rimane ancora abbastanza marginale

Directory:/ PHP512

SPRING FRAMEWORK 1.2.7

IL FRAMEWORK DEI FRAMEWORK

Spring è decisamente un argomento caldo. Lo abbiamo definito come “il framework dei framework” perché riunisce sotto un unico cappello una serie di tecnologie già esistenti, utilizzandole però nel migliore dei modi e fornendo allo sviluppatore un'interfaccia omogenea. Si tratta di un argomento caldo perché in moltissimi stanno adottando questo strumento per lo sviluppo delle proprie applicazioni, per l'alta flessibilità nell'utilizzo di oggetti di business e per la sua completezza. Sicuramente ne parleremo ancora su ioProgrammo

Directory:/ Spring

LAVADORA 1.0

UN PLUGIN PER SVILUPPARE WEB SERVICES CON ECLIPSE

Il progetto Lavadora nasce con lo scopo di creare un plugin per lo sviluppo di Web Services su piattaforma Eclipse. Al momento fornisce diversi wizard per lo sviluppo dei ws, generazione automatica del WSDL, browsing degli UDDI e deployment facilitato su Tomcat. Un progetto importante dunque per gli sviluppatori Java che desiderano pubblicare sul Web i loro servizi nell'ambito della creazione di applicazioni geograficamente distribuite.

Directory:/ Lavadora

ORDINAMENTI PER DISTRIBUZIONE

SI TRATTA DI METODI DALLE SORPRENDENTI PERFORMANCE CHE PERÒ RICHIEDONO UN'ATTENTA ANALISI DEI DATI SOTTOPOSTI A ORDINAMENTO. CERCHIAMO DI CAPIRE COME FUNZIONANO E A COSA BISOGNA STARE ATTENTI

Un'importante categoria di metodi di ordinamento è detta distribution sort, ossia ordinamento per distribuzione. Questa famiglia di algoritmi ribalta la logica sottesa ai più conosciuti metodi di sorting, come *bubble sort* o *quick sort*, per i quali è ripetuto "continuamente" il confronto tra elementi della lista da ordinare, che produce eventualmente scambi. Qui non sono previsti confronti, si tratta semplicemente di collocare (distribuire) appropriatamente i dati in aree di memoria; magari iterando il processo in più fasi. Sono stati introdotti molto tempo addietro, negli anni sessanta, nello stesso periodo che hanno visto la luce i più conosciuti metodi di ordinamento, nel tempio mondiale della scienza: il MIT. Attualmente gli algoritmi di ordinamento per distribuzione stanno trovando sempre più applicazioni per le loro formidabili prestazioni. In passato erano stati riposti nel cassetto per via della loro "considerevole" richiesta di risorse. Oggi con le memorie RAM che aumentano di giorno in giorno la loro capacità e velocità questo ostacolo è superato.

REQUISITI

Conoscenze richieste
Basi di programmazione C++

Software
-

Impegno

Tempo di realizzazione

RADIX SORT CON ESEMPIO

Radix sort è un veloce e stabile algoritmo di ordinamento che ordina un insieme di voci identificate in modo univoco da chiavi. Ogni chiave può essere una stringa o un numero. Il metodo è anche conosciuto come *postal sort*. L'algoritmo opera con complessità $O(n*k)$ con n il numero di elementi e k la lunghezza media delle chiavi. *Radix sort* funziona come segue:

L'ORDINAMENTO DEL POSTINO

È di fatto l'algoritmo usato negli uffici postali per l'ordinamento di pacchi e lettere. Si procede ordinando in sequenza lo stato, la provincia, la citta, la zona, la strada e il numero. Ha complessità O(k*n) come molti altri della famiglia, e solo in determinate condizioni (se k molto minore di n). Si differenzia dal radix sort perché procede secondo una metodologia top down (dal più grande al più piccolo). Viceversa radix sort è sviluppato in modalità buttom up.

- Considera la parte meno significativa (una singola cifra, un singolo o un gruppo di bit o un carattere) di ogni chiave;
- Ordina la lista degli elementi così selezionati, mantenendo l'ordine pregresso nel caso di parti meno significative identiche (ciò garantisce la caratteristica di stabilità dell'algoritmo);
- Ripete l'ordinamento per tutte le altre parti più significative delle chiavi.

Si ottiene complessità $O(n)$ per un numero di voci molto più grande della grandezza della chiave. Se ad esempio la chiave usata è uno short integer l'intero processo di ordinamento si esaurisce in due soli passaggi. Il problema che non sempre ci troviamo in questa situazione. Nel caso sfavorevole oltre ad una degenerazione del metodo bisogna fare i conti con un "eccessivo" uso della memoria. Esistono dei metodi ad istogramma che si applicano in questi casi e riducono tale spreco. Facciamo un esempio concreto. Supponiamo di voler ordinare la seguente sequenza di numeri:

150, 7, 25, 30, 12, 305, 46, 59

Dopo la prima iterazione si ordinano le cifre meno significative e si ottiene:

150, 30, 12, 25, 305, 46, 7, 59

Si nota come 150 è sempre prima di 30, così come 25 è riportato prima di 305, proprio perché si mantiene l'ordine che avevano nella lista in precedenza. Dopo la seconda fase si distribuiscono i numeri in base alla seconda cifra significativa. Ovviamente, si assume che numeri con una sola cifra siano interpretati con lo zero davanti. Quindi 7 è come se fosse 007 (da non confondere con l'agente segreto!).

305, 7, 12, 25, 30, 46, 150, 59

Infine, dopo la terza fase sorprendentemente la lista si ritrova ordinata:

7, 12, 25, 30, 46, 59, 150, 305

SOLUZIONE RICORSIVA

La soluzione naturale è ricorsiva, ma questa può essere anche implementata con l'uso di strutture supplementari in modo iterativo. Visioniamo la soluzione ricorsiva. Supponiamo di applicare l'algoritmo a numeri, facilmente si può posporre al caso di stringhe. Per passi l'algoritmo funziona così:

1. Prendiamo la cifra significativa di ogni chiave;
2. Ordiniamo la lista di elementi basata su quella cifra raggruppando elementi con la stessa cifra nello stesso secchio;
3. Ricorsivamente ordiniamo ogni secchio, partendo con la cifra immediatamente successiva in termini di significatività;
4. Concateniamo i secchi.

Nell'implementare la soluzione proposta si possono usare liste a puntatori che si prestano in modo naturale alla strutturazione dei secchi. Ogni bucket è un nodo che conterrà tutti gli elementi ad esso riferito ossia una lista ad essa appesa. Tale elaborazione rischia di innescare frammentazione della memoria che per grandi quantità di dati potrebbe degradare le performance dell'algoritmo. Una alternativa è usare degli array, strutture quindi che sono stabili in memoria perché allocate in fase di dichiarazione. In tal caso sicuramente si hanno computazioni più veloci che si pagano però con un maggiore spreco di memoria. Ad ogni modo se la tipologia dei dati lo consente è preferibile usare la seconda struttura proposta. Facciamo un esempio. Vogliamo ordinare la stessa sequenza precedentemente proposta.

150, 7, 25, 30, 12, 305, 46, 59.

Collochiamo nei primi secchi gli elementi con cifra più significativa. Delle centinaia:

zero: 7, 25, 30, 12, 46; uno: 150; tre : 350.

Distribuiamo adesso rispetto alla seconda cifra. È necessario farlo per il solo secchio zero delle centinaia: zero: 7; uno: 12; due: 25; tre: 30; quattro: 46; cinque: 59. Infine, bisogna ordinare per la cifra meno significativa. Non è necessario poiché non vi sono bucket che contengono più di un elemento. Concateniamo i risultati partendo dal bucket zero della cifra meno significativa computata. Si ottiene così l'elenco numerico ordinato:

7, 12, 25, 30, 46, 59, 150, 305

Questo versione è efficiente quando la maggior parte delle righe necessita di essere ordinata con poche cifre significative di profondità. In tal caso solo poche cifre poco significative necessitano di essere elaborate. Se la tipologia dei dati invece richiede ripetute esplorazioni in profondità è preferibile la versione non ricorsiva di radix sort. Partendo dalle cifre più significative, al contrario di come è stato descritto l'algoritmo in principio, effettivamente le prime cifre ad essere distribuite sono le meno significative. Ciò è dovuto al modo con cui viene sviluppata la ricorsione.

RADIX SORT NON RICORSIVO

Funziona come descritto al principio e fa uso di alcune duplicazioni delle strutture dati dove sono inizialmente contenuti i dati per consentire il travaso dei numeri che man mano vengono ordinati, nelle varie fasi della distribuzione da una strutture all'altra. Di seguito è riportato il codice che implementa radix sort. Si presuppone di passare come parametro una lista vettore dati di numeri *long* (interi lunghi) che contiene i numeri da ordinare. Si inseriscono inizialmente cento numeri casuali, attraversi la procedura *numeri_casuali*. La routine intermedia *radix_sort* è richiamata dalla *radix_sort_ric*. Per intenderci poiché si vogliono ordinare numeri interi *(long)* di quattro byte saranno necessarie quattro fasi (primo parametro di *radix_sort*), infatti la funzione è richiamata quattro volte. È necessario usare il comando >> di *shift* di un byte e il relativo cast esplicito. Con *memset* si inizializza il blocco (vettore *cont*) a zero.

```
void radix_sort (int byte, long N, long *s, long *d)
{ int i;
  long cont[256];
  long indice[256];
  // inizializza il vettore contatore a 0;
  memset (cont, 0, sizeof (cont));
  // shift di un byte e conversione esplicita che
  // permette l'analisi di cifredi significatività
  // incrementale
  for ( i=0; i<N; i++ ) cont[((s[i])>>
                                    (byte*8))&0xff]++;
  indice[0]=0;
  for (i=1; i<256; i++) indice[i]=indice[i-1]+cont[i-1];
  for ( i=0; i<N; i++ ) d[indice[((s[i])>>
                             (byte*8))&0xff]++] = s[i];}
void radix_sort_ric (long *s, long *temp, long N)
{   radix_sort (0, N, s, temp);
    radix_sort (1, N, temp, s);
    radix_sort (2, N, s, temp);
    radix_sort (3, N, temp, s); }
void numeri_random (long *dati, long N)
```

APPLICAZIONI PER PROCESSORI PARALLELI

La versione ricorsiva del radix sort ha importanti applicazioni nell'ambito del calcolo parallelo. Ogni suddivisione può essere elaborata indipendentemente dal resto. Così ogni ricorsione viene passata al primo processore disponibile. Un primo processore potrebbe essere usato per avviare l'algoritmo quindi elaborare la cifra più significativa. Un secondo, un terzo e così via sarebbero innescati nell'uso per le cifre immediatamente dopo significative.

```
{
   for ( int i=0; i<N; i++ ) dati[i]=rand()|(rand()<<16);
}
long dati[100];
long temp[100];
int main (void)
{ int i;
 numeri_random(dati, 100);
 radix_sort_ric (dati, temp, 100);
 for ( i=0; i<100; i++ ) cout << dati[i] << '\n';
 getch();
}
```

BUCKET SORT

Inizialmente gli elementi sono distribuiti in alcune aree, che chiameremo anche *secchi* o *bucket*. Queste aree sono disposte in modo che siano ordinate rispetto alle chiavi. Ogni *bucket* se necessario viene ordinato e i contenuti sono concatenati. Studiamo *Bit sort* che in letteratura si trova sia come sinonimo di *bucket sort* sia variante dello stesso. Qui si estremizza il concetto poiché ogni secchio contiene un solo elemento, o al più copie delle stesso elemento ripetuto nella lista da ordinare. Si assume che le chiavi delle voci che si desidera ordinare siano contenute in un "circoscritto" range. Verrà chiarito dall'esempio il significato dell'accezione circoscritto nell'ambito del nostro contesto. Inoltre, ci deve essere una sola voce per ogni chiave. Il secondo esempio ci mostrerà come sia facilmente superabile questo vincolo. L'algoritmo si sviluppa su tre fasi. La prima configura un elenco di binari in un insieme pari alla cardinalità delle chiavi. A tale scopo verrà usato un array di variabili booleane inizializzate a falso. La seconda fase esamina ogni voce e usa il valore della chiave per settare a vero l'array prima descritto. Ogni dato che si incontra setterà a vero il corrispondente indice del vettore di booleani. L'idea sta quindi nel far corrispondere un valore ad un indice. La terza prevede che si scandisca il vettore dei binari al fine di comporre il vettore di dati ordinato. Esaminiamo subito la complessità perché certamente provoca uno shock. Il *bin sort* è *O(n)*, esattamente lineare. Il prezzo da pagare però è che vi siano delle pre-condizioni alquanto restrittive. Se *n* è la dimensione del vettore da ordinare, o se preferite il numero di elementi da trattare, e con m indicheremo il massimo valore che può assumere una chiave. È facile intuire che la fase due necessita di *n* passaggi mentre la tre di *m*. Quindi si ha *O(n+m)*. Ora, se ci troviamo in una tipologia di dati per la quale *m* o è più piccolo di *n*, oppure è comunque comparabile, allora si ottiene il risultato ad effetto *O(n)* già enfaticamente annunciato. Purtroppo se i dati sono distribuiti su un campo di variazione molto ampio, per cui la chiave massima assume un valore molto grande ci troviamo nella situazione per la quale *m* è molto maggiore di *n*. In tal caso la complessità passa a *O(m)*. Facciamo una semplice stima. Se ad esempio desideriamo ordinare *10^4 (n)* numeri interi a 32 bit, e questi si distribuiscono in tutto il range definito dall'insieme, allora *m* è pari a *2^32*. Così oltre che una gran quantità di memoria anche l'algoritmo degenera. In questo caso sono preferibili algoritmi come *heap sort* o *quick sort*. Ricordo che per questi la complessità è circa *O(nlog(n))*. Facendo i conti risulta circa *nlog(n) 2^17* che è molto minore di *2^32*. Ecco l'implementazione, nel caso di ordinamento per un insieme di interi. Si può notare come siano sviluppate in modo naturale le tre fasi dell'algoritmo, e come adesso risultino ancora più chiare. Le costanti *MMAX* e *NMAX* contengono rispettivamente il valore massimo della chiave (in altri termini non possono essere trattati numeri più grandi di *MMAX*) e il massimo numero di elementi da ordinare. I vettori *d* contiene i dati, mentre *b* è di supporto e contiene i valori booleani. Infine, n è il reale numero di elementi da ordinare, ovviamente deve essere minore o uguale di *NMAX*.

```
// Costanti e variabili globali
const int MMAX=500;
const int NMAX=100;
int n;
int d[NMAX];
bool b[MMAX];
// BIN sort
void bin_sort (int *dati, bool *bin)
{   int i,j;
    // inizializzazione del vettore bin (1°)
    for (i=0; i<MMAX; i++) bin[i]=false;
    // Costruzione del vettore bin (2°)
    for (i=0; i<n; i++) bin[dati[i]]=true;
    // Ricomposizione del vettore dati (3°)
    j=0;
    for (i=0; i<MMAX; i++)
    if (bin[i]) dati[j++]=i;
};
int main()
{   input_dati(d);
    bin_sort(d,b);
    output_dati(d);
}
```

Per ragioni di spazio non vengono riportate le implementazioni delle routine standard di input e output, che troverete comunque sul CD. Il limite di tale algoritmo oltre che nella tipologia dei dati che devono sottostare ad un determinato range, sta nel fatto che non si possono avere ripetizioni di chiavi. Questo secondo inconveniente si può facilmente superare considerando anziché un vettore di booleani uno di interi. Inizializzato a zero. Così ogni qual volta si incontra un numero, che nel vet-

NOTA

ALBERO

I metodi di visita permettono di accedere a tutti gli elementi dell'albero, quelli maggiormente usati sono tre:

Anticipato
Nell'ordine vengono visitati:

- **la radice,**
- **il sottoalbero sinistro,**
- **il sottoalberodestro.**

Simmetrico
Nell'ordine vengono visitati:

- **il sottoalbero sinistro,**
- **la radice,**
- **il sottoalbero destro.**

Posticipato
Nell'ordine vengono visitati:

- **il sottoalbero sinistro,**
- **il sottoalbero destro,**
- **la radice.**

tore citato rappresenta l'indice, si incrementa il valore corrispondente. Alla fine il valore contenuto nelle singole celle del nuovo array modificato indicherà quante volte si ripete quella determinata voce. Si tratta di *Bucket sort*. Le modifiche da apportare rispetto al caso precedente sono poche. In fase di dichiarazione specificare che *b* è un vettore di interi:

```
const int MMAX=500;
const int NMAX=100;
int n;
int d[NMAX],b[MMAX];
```

è la procedura di ordinamento dovrà essere modificata come segue.

```
// BUCKET sort
void bucket_sort (int *dati, int *bin)
{   int i,j,z;
    // inizializzazione del vettore bin (1°)
    for (i=0; i<MMAX; i++) bin[i]=0;
    // Costruzione del vettore bin (2°)
    for (i=0; i<n; i++) bin[dati[i]]++;
    // Ricomposizione del vettore dati (3°)
    j=0;
    for (i=0; i<MMAX; i++)
    for (z=0; z<bin[i]; z++) dati[j++]=i;
};
```

come si può notare si apportano modifiche a tutte le fasi. La prima inizializza con zero anziché con *false*. La seconda incrementa la cella indicata dal valore del dato anziché portarla a *true*. È questo l'espediente che permette di considerare i duplicati di una generica chiave. Infine l'ultima, semplicemente con un secondo ciclo si aggiorna il numero di duplicati valori nel vettore dati. Non è stato previsto alcun confronto in conformità alla logica degli ordinamenti per distribuzione. Se non vi è alcun valore nella generica casella di *bin*, il ciclo di *for* non effettua alcun ciclo, non producendo alcuna assegnazione.

ALBERI INCREMENTALI

Un altro approccio per comprendere il metodo è tracciare le soluzioni attraverso la costruzione di un albero. Al termine del processo si dovrà semplicemente esplorare l'albero a partire dalle foglie per avere l'insieme dei dati ordinato. Per capire associamo la costruzione ad un esempio. Questa volta consideriamo come chiavi delle stringhe. Ecco una sequenza che può servire allo scopo. *Tartaruga, gatto, maiale, cane, topo, gallo, serpente*. L'albero che si costruisce è rappresentato in **Figura 1**.

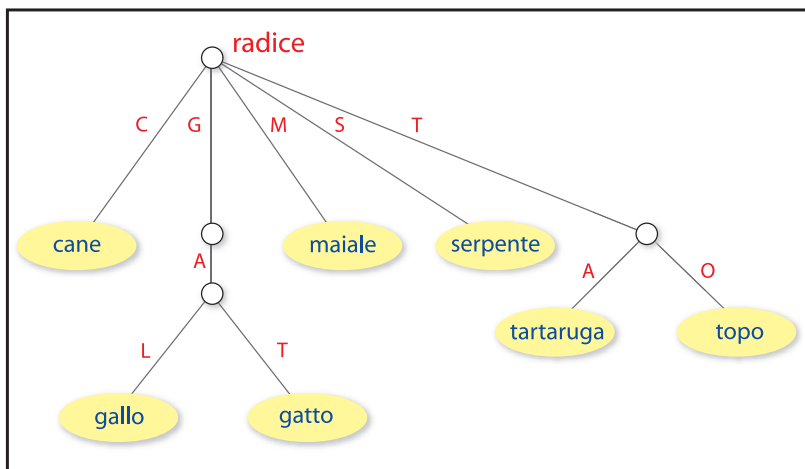



Fig. 1: ***Albero corrispondente all'esecuzione di radix sort su un insieme di stringhe***

Il *secchio c* contiene solo *cane* mentre il *secchio g* contiene a un altro secchio o che contiene le due voci *gallo* e *gatto*. Ottenere i dati ordinati, equivale a effettuare una visita dell'albero posticipato (come indicato in figura). Per terminare vediamo quale albero corrisponde alla sequenza numerica che abbiamo usato come esercizio.

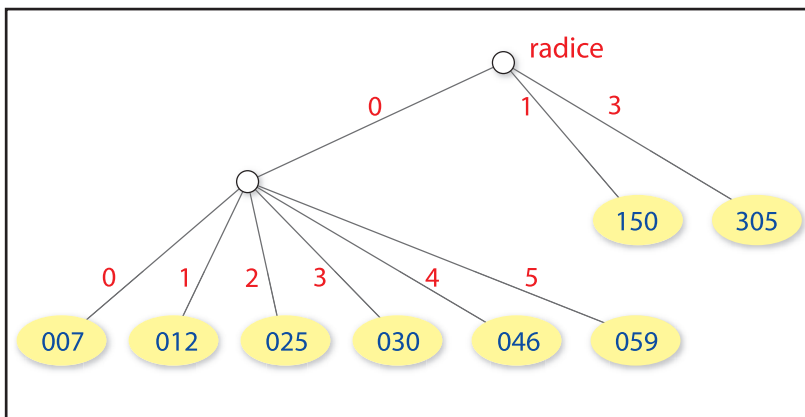



Fig. 2: ***Albero corrispondente alla esecuzione di radix sort su sequenza numerica***

CONCLUSIONI

Le performance degli algoritmi esaminati sono talmente formidabili che non richiedono ulteriori commenti. Sulle precondizioni, invece, vorrei puntualizzare che dipendono molto dall'ambito dove l'ordinamento deve essere svolto. Per alcune tipologie di dati questi algoritmi non hanno antagonisti, sono semplicemente i migliori. Non a caso sono parte integrante di molti motori 3-D.
Ambienti in cui la velocità è un vero elemento discriminante. In altri casi, invece, un'attenta analisi del programmatore (meglio dell'analista) deve portare ad optare per l'applicazione di altri metodi. Vi aspetto per il prossimo appuntamento per trattare ancora lo sconfinato mondo del sorting.

Fabio Grimaldi